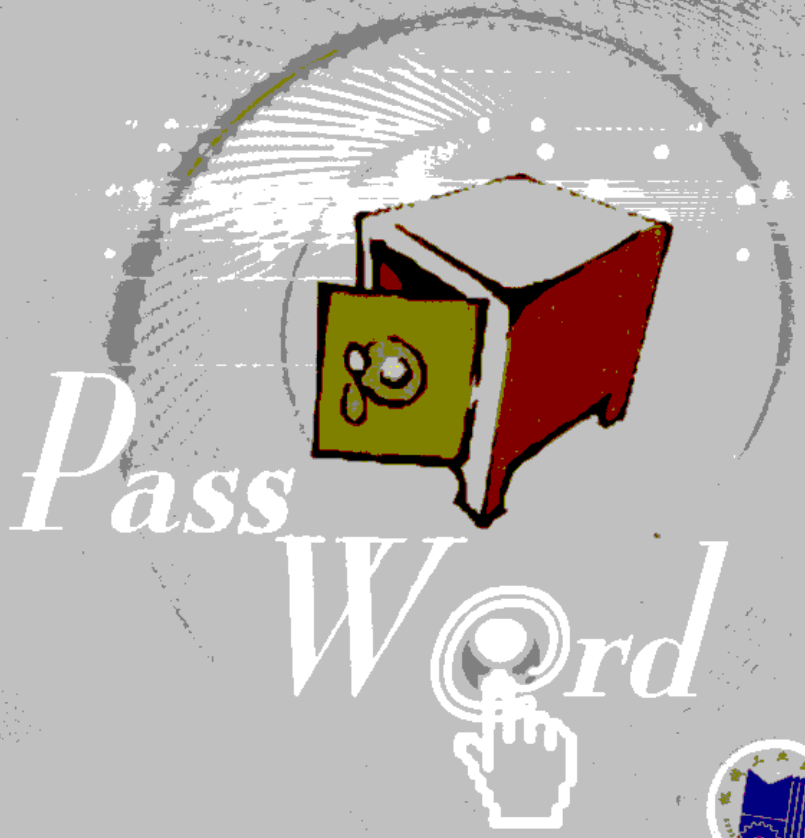
高等院校信息安全专业规划教材

现代密码技术

- 介绍基本的密码系统模型以及密码编码学、密码分析学的基本原理和方法
- 分析现代密码技术发展的各种理论基础
- 研究密码学原理，并对其安全性进行分析
- 展望密码技术未来的发展前景，指出可能的研究领域

李建华　主编
张爱新　马春波　陈恭亮　李生红　等参编



机械工业出版社
CHINA MACHINE PRESS

ISBN 978-7-111-21120-4

◎策划 胡毓坚

◎封面设计 旭洲企划 刘吉维

高等院校信息安全专业规划教材

- 信息安全概论
- 信息安全的数学基础
- 信息安全形式化基础
- 计算机网络安全技术
- 计算机系统安全原理与技术
- 密码理论
- 网络安全程序设计
- 网络信息安全
- 网络信息对抗
- 网络内容安全技术
- 计算机病毒与防范
- 信息安全基础设施
- 信息安全中的智能技术应用
- 信息系统安全测评
- 信息系统安全管理学
- 信息隐藏技术及应用
- 入侵检测与预警技术
- **现代密码技术**
- 无线局域网安全——方法与技术
- 信息安全工程学
- 密码协议形式化分析
- 信息安全系统设计与分析
- 电子对抗原理与技术
- 信息战理论与技术
- Internet 安全与工具

编辑热线：(010)88379739

地址：北京市百万庄大街22号 邮政编码：100037

联系电话：(010) 68326294 网址：http://www.cmpbook.com

E-mail:online@cmpbook.com

定价：20.00元

ISBN 978-7-111-21120-4

高等院校信息安全专业规划教材

现代密码技术

李建华　主编

张爱新　马春波　陈恭亮　李生红　等参编

机械工业出版社

密码技术是一门古老而又年轻的学科。本书概述了传统加密技术，简要介绍了公钥加密、私钥加密等现代密码基础，并在此基础上详细地叙述了现代密码技术的新进展，主要包括：旨在取代数据加密标准（DES）的先进加密标准——AES；建立在椭圆曲线上的公钥密码体制——椭圆曲线密码技术；基于物理学的密码体制——量子密码与混沌密码技术；基于生物技术的密码理论——DNA 密码技术。

本书可用作高等院校信息安全专业的教材，亦可作为高等院校电子信息类、计算机类等相关专业的参考资料。

图书在版编目（CIP）数据

现代密码技术/李建华主编．—北京：机械工业出版社，2007.3
高等院校信息安全专业规划教材
ISBN 978-7-111-21120-4

Ⅰ.现… Ⅱ.李… Ⅲ.密码－技术－高等学校－教材
Ⅳ.TN918.1

中国版本图书馆 CIP 数据核字（2007）第 032951 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）
策划编辑：赵　慧　责任编辑：戴　琳　责任校对：王　欣
责任印制：杨　曦
北京机工印刷厂印刷
2007 年 5 月第 1 版第 1 次印刷
184mm×260mm · 11 印张 · 267 千
0 001—4 000 册
标准书号：ISBN 978-7-111-21120-4
定价：20.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
销售服务热线电话：（010）68326294
购书热线电话：（010）88379639　88379641　88379643
编辑热线电话：（010）88379739
封面无防伪标均为盗版

出版说明

信息技术的发展和推广，为人类开辟了一个新的生活空间，它正对世界范围内的经济、政治、科教及社会发展各方面产生重大的影响。如何建设安全的网络空间，已成为一个迫切需要人们研究、解决的问题。目前，与此相关的新技术、新方法不断涌现，社会也更加需要这类专门人才。为了适应对信息安全人才的需求，我国许多高等院校已相继开设了信息安全专业。为了配合相关的教材建设，机械工业出版社邀请了解放军信息工程大学、解放军理工大学通信工程学院、上海交通大学、西安电子科技大学、湖南大学、南京邮电学院等高校的专家和学者，成立了教材编委会，共同策划了这套面向高校信息安全专业的教材。

本套教材的特色：

1. 作者队伍强。本套教材的作者都是全国各院校从事一线教学的知名教师和学术带头人，具有很高的知名度和权威性，保证了本套教材的水平和质量。

2. 系列性强。整套教材根据信息安全专业的课程设置规划，内容尽量涉及该领域的方方面面。

3. 系统性强。能够满足专业教学需要，内容涵盖该课程的知识体系。

4. 注重理论性和实践性。按照教材的编写模式编写，在注重理论教学的同时注意理论与实践的结合，使学生能在更大范围内、更高层面上掌握技术，学以致用。

5. 内容新。能反映出信息安全领域的最新技术和发展方向。

本套教材可作为信息安全、计算机等专业的教学用书，同时也可以供从事信息安全工作的科技人员以及相关专业的研究生参考。

机械工业出版社

前　言

随着信息技术的迅速发展，信息已成为当今社会的一种重要资源，计算机系统和通信网络的广泛应用，增加了人们在信息存储及信息交流中对这些系统的依赖性。随着信息资源的深入开发利用，政府管理、企业生产、社会公共服务、金融、财税等的信息化、网络化已在世界范围内迅猛展开。因此，信息系统中存储、处理和传送的信息越来越多地牵涉到个人的重要数据及国家的政治、军事、外交和经济机密。同时，由于计算机网络系统所具有的开放性、社会性和共享性，其安全性成为人们日益关注的问题。目前，黑客对计算机网络的攻击手段层出不穷，网络犯罪日趋严重，因而保护重要敏感信息免遭泄漏，确保系统免受网络攻击是极为必要的。一个现代信息系统若不含有信息安全技术措施，则不能被认为是一个完整的信息系统。在众多的信息保护和保密方法中，密码技术是保证信息安全的一种关键技术，因而密码学是现代信息理论和技术中一门不可或缺的学科。

密码学的发展可追溯到几千年前，经历了初级手工阶段、中级机械阶段和现在的电子阶段三个发展阶段。早期的密码技术发展缓慢，主要用于军事、外交、情报等敏感领域。香农（Shannon）于1949年发表的《保密系统的信息理论》一文为私钥密码系统提供了理论基础，标志着密码学从此成为一门科学。同时，微电子技术的发展使电子密码走上了历史舞台，尤其是20世纪70年代DES密码算法的公开发表及公开密钥思想的提出，都极大地促进了“现代密码体制”的发展。

密码学（Cryptology）包括两个相互对立、相互促进的分支学科：密码编码学（Cryptography）和密码分析学（Cryptanalysis）。前者研究使信息保密的技术和方法，一般是采用一组可逆的数学变换来达到隐藏信息的目的，从而保护信息不被第三方获悉；后者主要研究攻破一个密码系统的途径，以恢复被隐藏信息的本来面目。现代密码学的研究几乎涉及数学的各个分支，如信息论、数论、概率论、复杂性理论等；同时，密码学的研究还涉及物理学、生物学等多种学科的专业知识，如量子力学、混沌学、光学、DNA技术等。采用这些技术的优点在于，能够极大地提高密码系统的安全性，使之几乎在物理上“不可译”。目前，在各类专业书籍中，对基于数学的密码技术介绍得很多（椭圆曲线密码技术例外），而对基于物理学、生物学的密码技术的介绍则是一笔带过。随着电子芯片处理速度的提高，对原有密码系统的破译速度也大大提高，人们不得不靠增大密钥长度来维持密码系统的安全性，但这终究不是长久之计。据专业人士分析，若量子计算机研制成功，现有的基于数学理论的密码系统将毫无安全可言。

现代科学技术的迅猛发展，迫切需要一本全面论述现代密码技术的书籍。本书正是适应这种时代要求而产生的，其主要内容包括：

- 基本的密码系统模型以及密码编码学和密码分析学的基本原理和方法，这是现代密码技术发展的基础；
- 现代密码技术发展的各种理论基础，包括数论、离散对数问题、椭圆曲线原理、量子力学原理、混沌力学原理及DNA技术等；

- 基于上述理论的密码学原理，并对其安全性进行了分析；
- 综述了上述密码技术的应用实例，展望了其未来的发展前景，指出了可能的研究领域。

李建华教授担任本书主编，主持制订了编写大纲，并对全书进行统稿和修改。第1～3章主要由张爱新副研究员编写；第4、5章主要由陈恭亮教授编写；第6、7章主要由马春波博士后编写；李生红教授参与了第2、3、7章部分内容的编写；薛质教授、范磊副教授参与了第4章内容的讨论；王思叶硕士参与了第4章内容的编写；蒋兴浩副教授、刘功申副教授参与了第5章内容的讨论；李林森副教授、唐俊华副教授参与了第6章内容的讨论；李翔副教授参与了第7章内容的讨论。

本书的编写得到了曾贵华教授的大力支持和协助，曾教授对于教材定位、内容规划给出了具体的指导方案，并提供了第6章的研究资料，在此表示诚挚的谢意。

在本书的编写过程中，王士林老师、马进老师、萧海东博士、李谢华博士、拾以娟博士、刘杰博士、郝黎明博士、龚洁中硕士、赵诚明硕士、周建新硕士、王羽嘉硕士等协助收集、整理了本书部分资料，在此一并表示感谢。

由于编者水平有限，书中难免存在缺点和错误，殷切希望广大读者批评指正。

编　者

目　录

出版说明
前言
第1章　概论 …… 1
1.1　信息安全与密码技术 …… 1
1.1.1　背景 …… 1
1.1.2　信息安全基本要素 …… 1
1.1.3　密码学在信息安全中的重要作用 …… 3
1.2　密码技术概述 …… 3
1.2.1　密码系统模型 …… 4
1.2.2　密码编码学 …… 5
1.2.3　密码分析学 …… 5
1.3　传统密码技术 …… 6
1.3.1　置换密码 …… 6
1.3.2　替代密码 …… 7
1.3.3　转轮机 …… 11
1.4　小结 …… 11
1.5　习题 …… 11
第2章　现代密码技术基础 …… 13
2.1　相关基础知识 …… 13
2.1.1　信息论基础 …… 13
2.1.2　数学基础 …… 18
2.2　现代密码技术分类 …… 25
2.2.1　流密码与分组密码 …… 25
2.2.2　私钥密码与公钥密码 …… 37
2.3　数据加密标准 …… 38
2.3.1　DES算法描述 …… 39
2.3.2　DES算法的实现 …… 45
2.3.3　DES的安全性分析 …… 46
2.4　RSA密码算法 …… 50
2.4.1　RSA加密算法描述 …… 50
2.4.2　RSA数字签名 …… 51
2.4.3　RSA算法的实现 …… 52
2.4.4　RSA的安全性分析 …… 52
2.5　小结 …… 53
2.6　习题 …… 54

第3章 先进加密标准 …… 55
3.1 AES 简介 …… 55
3.2 MARS 密码算法 …… 56
3.2.1 MARS 算法描述 …… 57
3.2.2 MARS 安全性分析 …… 65
3.2.3 MARS 性能测试 …… 65
3.3 RC6 密码算法 …… 67
3.3.1 RC6 算法描述 …… 67
3.3.2 RC6 安全性分析 …… 69
3.3.3 RC6 性能测试 …… 69
3.4 Twofish 密码算法 …… 71
3.4.1 Twofish 算法描述 …… 71
3.4.2 Twofish 安全性分析 …… 76
3.4.3 Twofish 性能测试 …… 77
3.5 Serpent 密码算法 …… 80
3.5.1 Serpent 算法描述 …… 80
3.5.2 Serpent 安全性分析 …… 85
3.5.3 Serpent 性能测试 …… 85
3.6 Rijndael 密码算法 …… 85
3.6.1 Rijndael 算法描述 …… 86
3.6.2 Rijndael 安全性分析 …… 91
3.6.3 Rijndael 性能测试 …… 92
3.7 小结 …… 92
3.8 习题 …… 92
第4章 椭圆曲线密码 …… 93
4.1 基本概念 …… 93
4.1.1 群 …… 93
4.1.2 椭圆曲线上的加法规则 …… 94
4.1.3 建立在有限素数域上的椭圆曲线 …… 97
4.1.4 建立在有限二进制域上的椭圆曲线 …… 98
4.2 安全椭圆曲线 …… 99
4.2.1 计算椭圆曲线上点的个数 …… 99
4.2.2 求取椭圆曲线上的点 …… 99
4.2.3 构造一个有限域上的椭圆曲线 …… 100
4.3 建立在椭圆曲线上的密码体制 …… 100
4.3.1 椭圆曲线上的离散对数问题 …… 100
4.3.2 椭圆曲线密码算法 …… 101
4.3.3 椭圆曲线密码体制的安全性分析 …… 101
4.4 椭圆曲线密码体制的实现与应用 …… 102

4.4.1 椭圆曲线密码算法的软件实现 …… 102
4.4.2 椭圆曲线密码算法的硬件实现 …… 105
4.4.3 椭圆曲线 Diffie-Hellman 密钥交换 …… 106
4.4.4 椭圆曲线数字签名算法 …… 106
4.5 小结 …… 107
4.6 习题 …… 107
第 5 章 量子密码 …… 108
5.1 量子比特的基本属性 …… 108
5.1.1 量子叠加性 …… 109
5.1.2 单量子比特变换 …… 109
5.1.3 量子逻辑门 …… 110
5.1.4 Bell 理论与 EPR 纠缠态 …… 111
5.2 量子密码基本原理 …… 112
5.2.1 量子密码信息理论 …… 112
5.2.2 对敌手的检测理论 …… 113
5.2.3 保密加强理论 …… 114
5.3 量子密钥分发 …… 114
5.3.1 BB84 协议 …… 114
5.3.2 B92 协议 …… 115
5.3.3 E91 协议 …… 117
5.4 小结 …… 117
5.5 习题 …… 117
第 6 章 混沌密码 …… 119
6.1 混沌学基本原理 …… 119
6.1.1 控制混沌原理 …… 121
6.1.2 混沌同步原理 …… 124
6.2 混沌密码原理 …… 126
6.3 混沌密码的应用 …… 134
6.3.1 基于混沌分组密码的图像加密算法方案 …… 135
6.3.2 基于混沌的语音加密算法 …… 135
6.4 小结 …… 137
6.5 习题 …… 137
第 7 章 DNA 密码技术 …… 138
7.1 概述 …… 138
7.1.1 生物分子计算 …… 138
7.1.2 DNA 密码、传统密码和量子密码的比较 …… 139
7.2 DNA 密码学的理论基础 …… 139
7.2.1 粘贴模型的概念 …… 140
7.2.2 粘贴模型的物理实现 …… 143

7.3 DNA 加密技术 …… 146
7.3.1 使用随机一次性密码本的 DNA 密码系统 …… 146
7.3.2 使用替换的 DNA 加密系统 …… 146
7.3.3 DNA 异或一次性密码本密码系统 …… 148
7.4 DNA 加密技术的应用 …… 150
7.4.1 二维图像 DNA 密码系统 …… 150
7.4.2 DNA 隐写术 …… 152
7.5 展望 …… 157
7.6 小结 …… 159
7.7 习题 …… 159
附录 缩略语 …… 160
参考文献 …… 161

第1章 概 论

在本章中，将介绍信息安全与密码技术的一些基本概念，并以置换密码、替代密码等为例介绍传统密码技术的基本理论和学术思想。

1.1 信息安全与密码技术

信息安全与国家、企业、个人的利益息息相关，其重要性已得到了人们的普遍关注，成为世界各国的安全战略的重要组成部分。作为信息安全的核心技术之一，密码技术具有极为重要的基础支撑作用。

1.1.1 背景

在当今信息时代，信息已成为国家的重要战略资源。信息技术的迅速发展大大提高了社会生产力，引发了经济结构、社会结构和生活方式的深刻变革。自20世纪90年代以来，Internet日益普及和广泛使用，人类的活动、生活越来越依赖于信息网络的正常、安全运转。信息的安全直接关系到一个国家的政治稳定、经济发展和社会进步，可以毫不夸张地说，一个国家的信息安全程度将直接影响该国的综合国力和国际竞争力。没有信息安全，就没有完全意义上的国家安全。

同时，信息安全也与企业的发展、个人的利益息息相关。现在，任何个人或组织都可以通过网络与遍布在世界各个角落的计算机相联。人们可以与远在天涯海角的朋友聊天，给他们发送电子邮件；也可以利用网络实现远程教育和医疗、在线购物和付费等；通过网络，各企业组织可以向分布在世界各地的分支机构发号施令，与全球的商业伙伴谈判、签署合同，并且可以最大限度地搜索竞争对手的信息，密切关注他们的战略动态。在信息系统中存储、发送、处理的信息必然涉及到个人权益、企业生存、金融风险等各种机密信息。因此，信息安全在保障军事、政治、经济、外交等国家安全问题的同时，其商用价值和社会价值也得到了各界人士的普遍关注。

1.1.2 信息安全基本要素

信息安全是一个很广泛的范畴，它最基本的目标是：实现信息的机密性，保证数据的完整性，实现身份或信息的鉴别性，具有不可抵赖性，对信息进行授权和访问控制以及保证信息资源的可用性。有时人们也称之为安全六要素或一个信息系统应提供的六种安全服务。这些概念的提出源于现有的针对信息系统的攻击方式。

1. 安全攻击

正常情况下，系统中的信息流从一个主体（信息源）流到另一个主体（信息目的地），如图1-1所示。

通常对信息的攻击有4种方式：截获、中断、伪造、篡改，如图1-2所示。

图 1-2 中，A、B 分别表示正常的消息源和消息目的地，或者说是消息的发送者与接收者，I 表示网络攻击者。

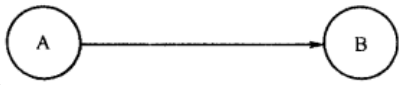


图 1-1　信息流的正常流向

由图 1-2a 可见，截获是指一未授权方（I）获取了对某一信息的访问，而 A 和 B 都没有意识到通信内容的泄漏。

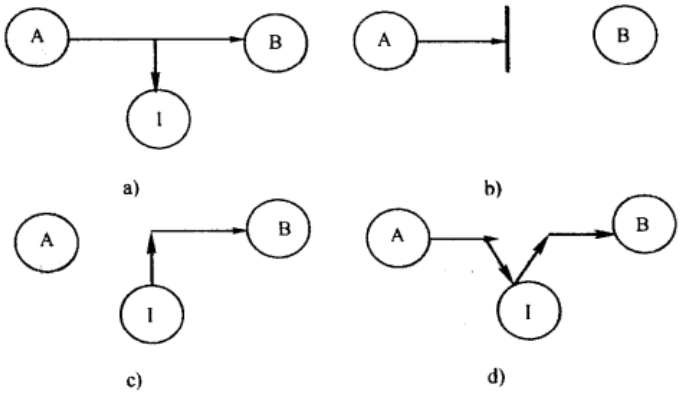


图 1-2　信息攻击

a）截获　b）中断　c）伪造　d）篡改

攻击实例如搭线窃听、非法复制文件等。这是对信息机密性的攻击。

图 1-2b 为中断方式，即信息资源被破坏或不能被利用，常见的如通信线路的切断、拒绝服务或文件系统的破坏等。这是对系统可用性的攻击。

图 1-2c 为伪造方式，即未授权方将伪造的信息插入系统中，A 意识不到有人在冒充他发送信息，而 B 认为该消息是 A 发送的。若 I 发送的消息是以前截获的，就是重放攻击。这些都是对鉴别性的攻击。

图 1-2d 为篡改方式，即未授权方不仅获得了对信息的访问控制，而且还能篡改这些信息。在图中，I 首先截获了 A 发送给 B 的消息，然后将修改后的消息发送给 B。这是对信息完整性的攻击。

可以看到，以上攻击方式中都含有对信息系统的授权及访问控制攻击。随着电子商务、电子政务的展开，对信息发送及接收的抵赖性攻击也受到了人们的日益关注。比如一个犯罪分子通过电子政务系统冒充政府发布命令，但事后否认这是他干的；或者一个商人通过网络获得了一笔转账，但他否认自己曾拿到过这笔钱，同样他也可以声称自己已寄出了一笔款项，而实际上并没有这样做。若没有很好的防范措施，这些都会极大地影响世界的政治秩序和经济秩序。

在形形色色的攻击方式中，若是对通信流进行偷听或监视以借此获得信息，就称为被动攻击，基本手段有信息内容分析、流量分析等；若涉及到对数据流的篡改或者产生新的虚假信息，则为主动攻击，比如伪装、重放、篡改消息、拒绝服务等。一般来说，对于被动攻击可以采取相应的措施防范，而完全防止主动攻击则是很难的。现今大部分的网络攻击都属于主动攻击。

2. 安全六要素

通过以上分析，可以知道，一个安全的信息系统必须具有以下六种安全要素并且提供相应的安全机制：

1）机密性：保护信息不会暴露给未授权的实体，这主要用以防止被动攻击。常用的技术手段是进行加密变换，使得信息从表面上无法识别，只有合法用户才能够将密文还原为明文，而非法的接收者是得不到真实的信息的。

2）完整性：确定信息没有被非法更改、删除、重放、延迟等。一般采用各种形式的消息认证码、杂凑函数等技术。

3）鉴别性：确保通信的可信性，主要包括两方面的内容：一是对通信双方身份的鉴别，以保证这两个实体身份的可信性，也即实体认证；二是确定消息来源的合法性，向接收方保证该消息确实来自它所宣称的消息源，即通常所说的数据源认证。

4）不可抵赖性：有时也称为不可否认性，主要是保护通信实体免遭系统中其他合法用户的攻击，防止发送方或接收方否认所传输的消息。在具有不可抵赖性的系统中，接收方能够证实该消息的确是由所宣称的发送方发送的，发送方也有证据证明所发送的消息的确是由所宣称的接收方接收的。

5）授权与访问控制：采用授权及访问控制可以限制或控制经通信链路对任何信息资源的访问能力。

6）可用性：可用性指的是经过授权的用户可以得到系统资源，并且享受到系统提供的服务。

1.1.3 密码学在信息安全中的重要作用

无论一个信息系统多么复杂，通信实体间的交流都是按照一定的规则进行的，这些规则称为通信协议。因此，信息系统的安全性除去物理因素（如硬盘损坏、遭受雷击、突然断电等）、管理因素（部门内部安全管理混乱等）、病毒影响等外部干扰外，主要就取决于协议的安全性。从上述安全六要素分析得出，一个安全协议最基本的属性有两个：一是保密性，另一个就是认证性。只要保证了这两个安全属性，通过协议的合理设计及其他一些技术手段就能使另外的安全要素得以实施。而在这两个属性中，保密性又是基础。没有保密性，认证性就无从谈起，因此现有的安全协议都无一例外地利用密码技术实现开放网络环境下的安全通信，这也是为什么将安全协议称为密码协议的原因。因此，密码技术是保证信息系统安全的重要技术，密码学是信息安全科学的一门基础科学。

1.2 密码技术概述

密码学是一门古老的科学，它的起源可以追溯到4000多年前的古埃及、巴比伦、古罗马和古希腊，主要经历了以下发展历程：

第一阶段从古代到1949年。这个时期的密码技术只能称为艺术而不是科学，密码的设计和分析是凭直觉和经验来进行的，而不是靠严格的理论证明。

第二阶段从1949年到1975年。香农于1949年发表的“保密系统的信息理论”一文，使密码学从此成为一门科学。但这一时期的密码学研究仍以秘密进行为主，其理论研究工作进展不大，公开发表的文献很少，密码技术仍然主要用于军事、政治和外交等机密领域。

到此为止，密码学的研究仍是很敏感的，人们形象地将其称为“墙边的花朵”，意思是说，虽然已有了丰硕的成果，但仍未盛开。

第三阶段从1975年至今，DES算法的公开发表以及Diffie-Hellman公钥密码思想的提出，在密码学的发展中具有里程碑的作用。期间，微电子技术、计算机技术、通信技术的发展也为密码学的研究、应用起了巨大的推动作用。

密码学又是一门年轻的科学。随着科学技术的进步，密码学的研究也日新月异。首先，密码学越来越依赖于数学知识，现代密码学离开数学几乎是不可想像的；同时，密码学还与其他多种学科相互渗透，如量子力学、光学、混沌学、生物学等，并且互相促进。

1.2.1 密码系统模型

密码学以研究秘密通信为目的。在一个密码系统中，通常将最初要发送的消息称为明文，经过秘密变换后的消息称为密文。消息的隐藏过程称为加密，其逆过程即把密文转变为明文的过程称为解密。加密时所采用的规则为加密算法，相应地，解密规则称为解密算法。一般而言，加密和解密操作都在密钥的控制下进行，分别称为加密密钥和解密密钥。通常，密码系统也称为密码体制。

综上所述，一个密码系统主要由以下五部分构成：

1）明文空间 M：所有明文的集合；

2）密文空间 C：全体密文的集合；

3）密钥空间 K：全体密钥的集合，其中每一个密钥 k 均由加密密钥 K_e 和解密密钥 K_d 组成，即 $K=(K_e, K_d)$，在某些情况下 $K_e=K_d$；

4）加密算法 E：一组以 K_e 为参数的由 M 到 C 的变换，即 $C=E(K_e, M)$，可简写为 $C=E_{K_e}(M)$；

5）解密算法 D：一组以 K_d 为参数的由 C 到 M 的变换，可表示为 $M=D(K_d, C)$ 或 $M=D_{K_d}(C)$。

可以将一个密码系统模型表示为图1-3。

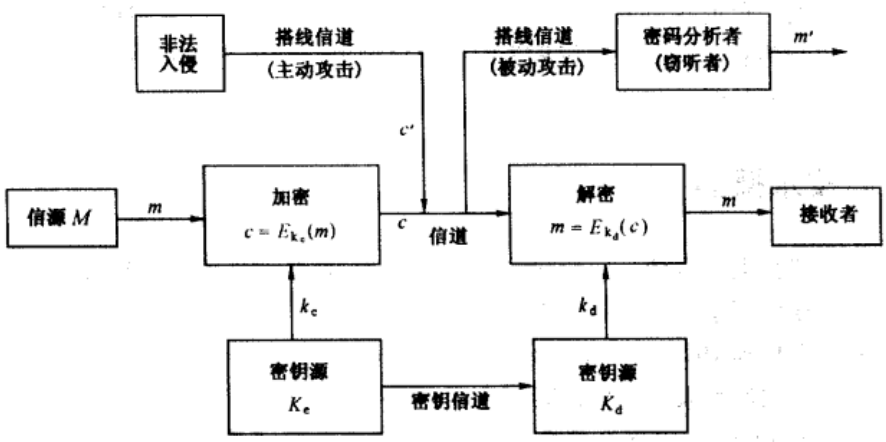


图1-3 密码系统模型

由图中看到，在一个密码系统中有两个互相对立的部分：一部分要保证消息的保密性和

认证性，而另一部分却试图寻求加密消息的破译或消息的伪造。由此形成了密码学的两个分支：密码编码学和密码分析学。它们相辅相成、相互对立而又相互促进地共同发展。

1.2.2 密码编码学

密码编码学是对消息进行编码以隐藏消息的一门学问。通常可以根据以下三个独立的方面对密码编码系统分类：

1）根据将明文转换为密文操作的类型，加密算法可分为替代和置换，前者是将明文中的每个元素映射为另一个元素，后者指明文中的元素被重新排列。有关内容将在1.3节中结合传统密码技术进行分析。

2）根据所使用的密钥的数量，可将密码编码系统分为单钥加密系统和双钥加密系统。如果发送方和接收方使用的密钥相同，即加密密钥与解密密钥相同或从其中一个很容易导出另一个，则该系统就是单钥系统，通常也称为对称加密、秘密密钥加密或常规加密；反之就是双钥系统，一般也叫非对称加密或公开密钥加密，可分别用图1-4、图1-5表示。

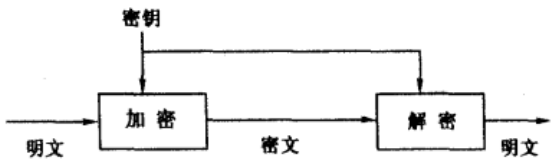


图1-4 单钥密码编码系统

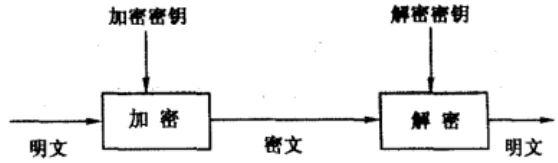


图1-5 双钥密码编码系统

3）根据对明文处理方式的不同，可将加密算法分为分组加密和流加密。分组加密一次处理一个输入块，对每个输入块产生一个输出块；流加密则是连续地处理输入元素，一般一次仅加密一个字符或位。

1.2.3 密码分析学

与密码编码学相对，密码分析学主要用于分析、破译密码，一般是在不知道密钥的情况下，恢复出明文。通过密码分析也可以发现密码体制的弱点。我们在讨论密码分析时，总是遵从柯克霍夫（Kerckhoff）假设，即认为密码分析者已经有了密码算法及其实现的全部详细资料，换句话说，我们考虑的是密码体制在最坏情况下的安全性。

对密码进行分析的尝试称为攻击。根据密码分析者可利用的数据，常见的密码分析攻击有四类，由弱到强，分别是惟密文攻击、已知明文攻击、选择明文攻击和选择密文攻击。

1）惟密文攻击：密码分析者有一些用同一密钥加密的密文，他们试图恢复出尽可能多的明文，或者推算出加密密钥以解出更多的密文。

2）已知明文攻击：密码分析者不仅得到了一些明文，而且也知道相应的密文，他们的任务是据此推出加密密钥或算法，而该算法可以对用同一密钥加密的任何密文进行解密。

3）选择明文攻击：密码分析者不仅可得到一些消息的密文和相应的明文，而且也可选择被加密的明文。通过选择特定的明文进行加密，有可能产生更多的关于密钥的消息。这比已知明文攻击更有效。如果分析者不仅能选择被加密的明文，还能基于以前的结果修正这个选择，就是自适应选择明文攻击。

4）选择密文攻击：密码分析者可选择不同的密文，并可得到对应的明文。这种攻击主要用于公钥算法。

根据密码分析者攻击密码的方法，可将密码分析攻击分为以下各类：

1）穷举攻击：对截获的密文，密码分析者试遍所有的密钥，以期得到有意义的明文；或者使用同一密钥，对所有可能的明文加密直到得到的密文与截获的密文一致。穷举攻击也称为强力攻击或完全试凑攻击。

2）统计分析攻击：密码分析者通过分析明文与密文的统计规律，从而得到它们之间的对应关系。

3）数学分析攻击：密码分析者根据加密算法的数学依据，利用数学方法（如线性分析、差分分析及其他一些数学知识）来破译密码。

在后续各章中，将针对不同的密码算法，分析相应的攻击方式，陆续讨论各种分析方法的实际应用。

1.3 传统密码技术

本节给出的传统密码技术是手工时代和机械时代的产物，现已被证明是不安全的，但作为现代密码的基础，了解它们的设计思想仍是很有意义的。

1.3.1 置换密码

置换密码是指将明文中的字母顺序重新排列，但字母本身不变，由此形成密文。换句话说，明文与密文所使用的字母相同，只是它们的排列顺序不同。

例如可以将明文按矩阵的方式逐行写出，然后再按列读出，并将它们排成一排作为密文，列的阶就是该算法的密钥。在实际应用中，人们常常用某一单词作为密钥，按照单词中各字母在字母表中出现的顺序排序，用这个数字序列作为列的阶。

【例 1-1】 以 coat 作为密钥，则各字母的排列顺序为 2、3、1、4，对明文“attack postoffice”的加密过程如图 1-6 所示。

密钥	c	o	a	t
阶	2	3	1	4
	a	t	t	a
	c	k	p	o
	s	t	o	f
	f	i	c	e

图 1-6 对明文“attack postoffice”的加密过程

按照阶数由小到大，逐列读出各字母，所得密文为：

tpocacsftktiaofe

对于这种列变换类型的置换密码，密码分析很容易进行：将密文逐行排列在矩阵中，并依次改变行的位置，然后按列读出，就可得到有意义的明文。为了提高它的安全性，可以按同样的方法执行多次置换。例如对上述密文再执行一次置换，就可得到原明文的二次置换密文为：

ostftatapckocfie

还有一种置换密码采用周期性换位。对于周期为 r 的置换密码，首先将明文分成若干组，每组含有 r 个元素，然后对每一组都按前述算法执行一次置换，最后得到密文。

【例 1-2】　一个周期为 4 的换位密码，密钥及密文同上例，加密过程如图 1-7 所示。

密钥	c	o	a	t	c	o	a	t	c	o	a	t	c	o	a	t
阶	2	3	1	4	2	3	1	4	2	3	1	4	2	3	1	4
明文	a	t	t	a	c	k	p	o	s	t	o	f	f	i	c	e
密文	t	a	t	a	p	c	k	o	o	s	t	f	c	f	i	e

图 1-7　周期性换位密码

1.3.2　替代密码

在讨论替代密码之前，先来看一下罗马皇帝 Julius Caesar 在公元前 50 年左右所使用的“凯撒密码”。他是将字母按字母表中的顺序循环排列，将明文中的每个字母用后面的第 3 个字母代替以得到对应的密文。

以英文字母表为例，凯撒密码所使用的明文字母表和密文字母表分别为：

明文字母表：a b c d e f g h i j k l m n o p q r s t u v w x y z

密文字母表：d e f g h i j k l m n o p q r s t u v w x y z a b c

那么，对于明文“attack postoffice”，经凯撒密码变换后的密文为“dwwdfnsrvwriilfh”。

凯撒密码可以说是替代密码的最简单的例子。在替代密码中，密文中的字母顺序与明文中的字母顺序一致，只是各密文字母是由相应的明文字母按某种映射变换得到的。

按照映射规则的不同，替代密码可分为三种：单表替代密码、多表替代密码和多字母替代密码。

1. 单表替代密码

单表替代密码对明文中的所有字母都用一个固定的明文字母表到密文字母表的映射 $f:M \to C, f(m_i) = c_i$。换句话说，对于明文 $(m_0, m_1, \cdots, m_{n-1})$，相应的密文为 $(c_0, c_1, \cdots, c_{n-1}) = (f(m_0), f(m_1), \cdots, f(m_{n-1}))$。

下面介绍几种简单的替代密码。

（1）加法密码

在加法密码中，映射规则可表示为：

$$f(m_i) = m_j, j = (i + k) \bmod n, 0 < k < n$$

其中 k 为密钥加密算法就是 $E_k(i) = (i + k) \bmod n$。

例如，可以将英文的 26 个字母分别对应于整数 0 ~ 25，则 $n = 26$，对应关系如表 1-1。

表 1-1　英文字母对照表

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

加法密码也称为移位密码，凯撒密码就是 $k=3$ 的加法密码。

【例 1-3】 取密钥 $k=9$，明文同例 1-1，则转换为密文的过程如下：

首先将其转化为数字序列：

0 19 19 0 2 10 15 14 18 19 14 5 5 8 2 4

然后每个数值加 9，并作模 26 运算，得到以下序列：

9 2 2 9 11 19 24 23 1 2 23 14 14 17 11 13

再将其转化为英文字母，可得密文：jccjltyxbcxoorln。

（2）乘法密码

乘法密码的映射规则可表示为：

$$f(m_i) = m_j,\ j = (ik) \bmod n,(k,n) = 1$$

其中 k 为密钥，加密算法就是 $E_k(i) = (ik) \bmod n$。

【例 1-4】 密钥及明文同例 1-3，采用乘法密码后的密文为：appasmfwgpwttusk。

乘法密码有时也叫做采样密码。

（3）仿射密码

同时运用加法密码和乘法密码，就构成了仿射密码，可以表示为：

$$f(m_i) = m_j,\ j = (ik_1 + k_0) \bmod n,(k_1,n) = 1,0 < k_0 < n$$

其中，$(k_0,\ k_1)$ 为密钥，加密算法可表示为 $E_k(i) = (ik_1 + k_0) \bmod n$。

解密算法是加密算法的逆变换，为 $D_k(j) = k_1^{-1}(j - k_0) \bmod n$。例子从略。

（4）多项式密码

仿照仿射密码，可以构造出更复杂的多项式密码：

$$E_k(i) = (i^t k_t + i^{t-1} k_{t-1} + \cdots + ik_1 + k_0) \bmod n$$

其中，$(k_i,n) = 1,i = 1 \cdots t,0 < k_0 < n$。

前三种密码都可以看作是多项式密码的特例。

（5）密钥短语密码

密钥短语密码是对上述各密码算法的改造，基本思想是任意选择一个短语作为密钥，去掉该密钥中的重复字母，并将它们依次写在明文字母表下，然后将明文字母表中从未在密钥短语中出现的字母依次写在该短语的后面，从而构造出一对明文、密文对照表。

【例 1-5】 取密钥短语为 good worker，去掉重复字母，得 godwrke，构造明/密文对照表如下：

明文表：a b c d e f g h i j k l m n o p q r s t u v w x y z;

密文表：g o d w r k e a b c f h i j l m n p q s t u v x y z;

那么对于例 1-1 的明文，其密文为：gssgdfmlqslkkbdr。

可以发现，采用以上方法，若密钥短语选择不合适，会造成大部分的密文字母与明文字母一致的现象，使得保密程度下降。可以结合置换密码的思想予以改进。

【例 1-6】 密钥短语同上，可以构造矩阵：

g o d w r k e

a b c f h i j

l m n p q s t

u v x y z

若按列读出，则可得明/密文对照表：

明文表：a b c d e f g h i j k l m n o p q r s t u v w x y z;

密文表：g a l u o b m v d c n x w f p y r h q z k i s e j t。

在单表替代密码中，对于多项式密码及其特例，由于它们的密钥量比较小，可以利用穷举攻击进行破译，尤其在计算机的帮助下，破译起来可以说是轻而易举的。而对于密钥短语替代密码，密文字母表本质上是明文字母表的一种排列，若字母表中有 n 个字母，可能的密文字母表是 $n!$ 种。若 n 较大，即使有计算机的帮助，穷举攻击也是不大现实的。即便如此，密码分析者利用统计分析方法，仍能迅速地攻破。下面简单地介绍一下统计分析攻击的基本思路。

任何自然语言都有其固有的统计规律性，如果明文语言的统计规律在密文中有所反应，则密码分析者就可以通过分析明文和密文的统计规律而破译密码。

比如，人们分析了英语的单字母、双字母及三字母的统计特性：

1）英文字母频度分类：

极高频度字母：e;

次高频度字母：t a o i n s h r;

中等频度字母：d l u c m;

低频度字母：p f y w g b v;

次低频度字母：j k q x z。

2）频度高的双字母组：th he in er an re ed on es st en at to nt ha nd ou ea ng as or ti is et it ar te se hi of。

3）频度高的三字母组：the ing and her ere ent tha nth was eth for dth hat she ion int his sth ers ver。

当密码分析者要对截获的密文进行分析时，首先统计密文中的字母出现频率，并与明文字母统计表比较。例如在英文中，字母 e 的出现频率远远高于其他字母，所以若一个密文字母出现频率极高，就可以断定该密文的对应明文是 e。进一步比较密文和明文的其他统计数据及分布模式，就可以确定出密钥，进而攻破单表替代密码。

2. 多表替代密码

由以上分析知道，在单表替代密码中，明文中的字母与密文中的字母一一对应，明文的统计规律能够在密文中反映出来，从而导致单表替代密码很容易被攻破。提高密码强度的一个办法就是采用多个密文字母表，使明文中的每一个字母有多种替代可能。这种密码称为多表替代密码。

一般来说，多表替代密码是用两个以上的替代表依次对明文消息的字母进行替代的加密方法。如果存在无限多个替代表，则称相应的密码为非周期多表替代密码，在这类密码中，对每个明文字母都采用不同的替代表进行加密，也就是说，每次加密的密钥都不同，因此也称为一次一密密码。这是惟一一种理论上不可破的密码，但在实际系统中是很难实现、推广的。

在实际应用中多采用周期性多表替代密码，即替代表个数有限，重复使用，可以描述为：

对于替代序列 $f=(f_1,f_2,f_3,\cdots,f_d)$，相应于明文 $M=(m_1,m_2,\cdots,m_{2d},\cdots)$ 的密文为：

$$C = (c_1, c_2, \cdots, c_{2d}, \cdots) = (f_1(m_1), f_2(m_2), \cdots, f_d(m_d), f_1(m_{d+1}), f_2(m_{d+2}), \cdots f_d(m_{2d}), \cdots)$$

【例 1-7】 仍取明文及密钥如例 1-1，则密钥对应的数字序列为（2，14，0，9），明文数字序列为（0，19，19，0，2，10，15，14，18，19，14，5，5，8，2，4），然后将明文分组，每组含有 4（密钥长度）个元素，最后明文数字依次与加密密钥进行模加运算（模为 26），过程为：

明文：0 19 19 0　　2 10 15 14　　18 19 14 5　　5 8 2 4

密钥：2 14 0 9　　2 14 0 9　　2 14 0 9　　2 14 0 9

密文：2 7 19 9　　4 24 15 23　　20 7 14 14　　7 22 2 13

将密文转化为英文字母，则相应的密文串为：chtjeypxuhoohwcn。

解密与加密类似，只是要进行模 26 减运算。

例 1-7 的密码就是维吉尼亚密码，典型的多表替代密码还有弗纳姆密码、博福特密码等，介绍从略。

3. 多字母替代密码

以上所介绍的替代密码每次只变换一个明文元素，如果每次加密两个以上的元素，就是多字母替代密码。它的优点是能够将字母的自然频度隐蔽或均匀化，从而较好地抵抗统计分析攻击。可以利用矩阵变换来描述多字母替代密码，因此多字母替代密码也称为矩阵变换密码。

设明文字母表为 Z_q，若采用 L 个字母为单位进行变换，则多字母替代是映射 $f: Z_q^L \rightarrow Z_q^L$。若 f 是线性变换，则可用一个 Z_q 上的 $L \times L$ 阶矩阵 $\boldsymbol{K}$ 表示，$\boldsymbol{K}$ 为密钥。若 $\boldsymbol{K}$ 是满秩的，则 $\boldsymbol{K}$ 是可逆的，存在逆变换 $\boldsymbol{K}^{-1}$。将 L 个字母的数字表示为 Z_q 上的一个向量 $m = (m_1, m_2, \cdots, m_L)$，则加密所得密文为

$$c = (c_1, c_2, \cdots, c_L) = m\boldsymbol{K} \bmod n$$

解密变换为 $m = c\boldsymbol{K}^{-1} \bmod n$

【例 1-8】 对于英文字母，明文空间及密文空间均为 Z_{26}，设密钥为 $\boldsymbol{K} = \begin{pmatrix} 4 & 9 \\ 3 & 7 \end{pmatrix}$，则 $\boldsymbol{K}^{-1} = \begin{pmatrix} 7 & -9 \\ -3 & 4 \end{pmatrix} \bmod 26 = \begin{pmatrix} 7 & 17 \\ 23 & 4 \end{pmatrix}$，加密明文 girl。其数字序列为（6，8，17，11），将明文分成两组：

$$m = (6 \quad 8), m' = (17 \quad 11)$$

则 $c = (6 \quad 8) \begin{pmatrix} 4 & 9 \\ 3 & 7 \end{pmatrix} \bmod 26 = (22 \quad 6)$

$$c' = (17 \quad 11) \begin{pmatrix} 4 & 9 \\ 3 & 7 \end{pmatrix} \bmod 26 = (23 \quad 22)$$

即密文数字序列为（22，6，23，22），所以密文字符串为 wgxw。

类似地，通过解密密钥 $\boldsymbol{K}^{-1} = \begin{pmatrix} 7 & 17 \\ 23 & 4 \end{pmatrix}$，可以将密文 wgxw 恢复为明文 girl。计算过程

从略。

多字母替代密码能够较好地抵抗频率攻击，一般采用惟密文攻击是很难凑效的，但它却不能抵抗已知明文攻击。例如，若密码分析者已知 L，且他也至少有 L 个不同的 L 元组 $M_i = (m_{1i}, m_{2i}, \cdots, m_{Li}), C_i = (c_{1i}, c_{2i}, \cdots, c_{Li}), 1 \leqslant i \leqslant L$，则攻击者可以定义两个 $L \times L$ 矩阵 $\boldsymbol{M} = (m_{ij})_{L\times L}, \boldsymbol{C} = (c_{ij})_{L\times L}$，则有 $\boldsymbol{C} = \boldsymbol{MK} \bmod 26$。若 $\boldsymbol{M}$ 可逆，可计算出 $\boldsymbol{K} = \boldsymbol{M}^{-1}\boldsymbol{C} \bmod 26$，从而破译该密码。若 $\boldsymbol{M}$ 不可逆，就要重新试另外 L 个明文/密文对。

1.3.3 转轮机

转轮机是能够自动处理加密的机械设备，它采用了多表替代的思想，实际上是一个长周期的多表替代密码机。

转轮机有一个键盘和一系列转轮，键盘用来输入明文字符串，而每个转轮是字母的任意组合，有 26 个接线端，用来完成一种简单的替代，同时每个转轮的输出端连到相邻转轮的输入端。这样当输入一个明文字母时，相应的信号从第一个转轮的输入端进入，依次经过各个相邻的转轮，最后从转轮组的输出端以密文形式输出。初始密钥由转轮和它们的起始位置决定，每输入一个字母，就可以使一个或多个转轮移动到一个新的位置，因而改变了密钥。对于有 n 个转轮的机器，其周期为 26^n。

比较著名的转轮机有德国的 Enigma、美国的 Sigaba、日本的 Red 和 Purple 等，它们都在第二次世界大战中发挥了重要作用。

1.4 小结

作为全文的引论，本章介绍了信息安全及密码学领域的一些基本概念，结合密码学发展过程中的早期成果，探讨了密码编码及密码分析的基本思路。后续各章将在此基础上继续深入、全面地介绍现代密码技术的基本原理及应用实例。

1.5 习题

1. 采用密钥“computer”对明文“a convenient way to express the permutation”进行置换加密。

2. 采用密钥“sorcery”对明文“laser beams can be modulated to carry more intelligence than radio waves”进行置换加密。

3. 用密钥 19 对“we hold these truths to be self evident”进行加法加密。

4. 在加法密码中，向前移位 t 步与向后移位 $26 - t$ 步是一样的吗？为什么？

5. 一密文信息“wdnhphrxwwrwkhedoojdph”是采用 Caesar 密码加密后得到的，试对其进行解密。

6. 为采用加法密码加密后的密文消息“jyribyrjgivkkpkvvky”解密。

7. 设有一乘法密码是在英文字母表上进行变换的，取 $k = 9$，试写出对应于明文消息“meet me at midnight”的密文信息。

8. 设有一多项式密码是在英文字母表上进行变换的，取 $k=\begin{pmatrix}17 & 17 & 5\\ 21 & 18 & 21\\ 2 & 2 & 19\end{pmatrix}$，试写出对应于明文消息“friday”的密文信息。

9. 设有一维吉尼亚密码是在英文字母表上进行变换的，取 $k=$ radio，对明文 m = “pay more money”进行加密。

10. 设有一多字母替代密码是在英文字母表上进行变换的，已知加密密钥为 $k=\begin{pmatrix}8 & 6 & 5 & 10\\ 6 & 9 & 8 & 6\\ 9 & 5 & 4 & 11\\ 5 & 10 & 9 & 4\end{pmatrix}$，截获一密文为 $c=$ jcomzlvbdvieqmxc，试破译出相应的明文。

第 2 章　现代密码技术基础

现代密码学的研究与分析涉及到众多学科的专门知识，例如数学、物理学、电子学、系统工程、语言学等。本章将简要介绍与后续各章关系较为密切的一些基础知识，主要包括信息论与密码学、密码学中的数学知识、相关物理学与生物学知识。同时从不同的角度对现代密码技术进行分类，分析其基本思想。最后介绍两个重要的密码算法：DES 和 RSA。

2.1　相关基础知识

本节主要介绍与现代密码技术相关的一些基础知识。

2.1.1　信息论基础

现代信息论是由香农（Shannon）于 1948 年首先确立的，他在论文“通信的数学理论”中详细阐明了如何用信息论的观点处理存在随机干扰的通信系统中的信息传输问题。他将通信系统表示为图 2-1 所示的模型。

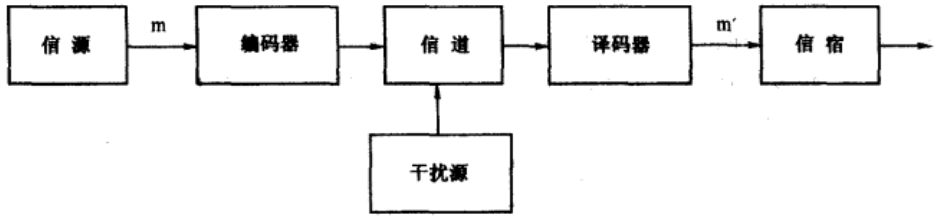


图 2-1　通信系统模型

在存在干扰的情况下，信源发送的消息 m 到达信宿时变为 m'，接收者就是要从 m'中恢复出原来的消息。为此，发送者要对消息进行编码，而接收者要对消息进行译码，其目的就是在有扰信道中使信息无错或差错尽可能小地传输。

香农也最早将信息论的观点用于密码学的研究，他在 1949 年发表了“保密系统的通信理论”一文，宣告了科学的密码学信息理论时代的到来，为私钥密码学奠定了理论基础。保密系统可表示为图 2-2 所示的模型。

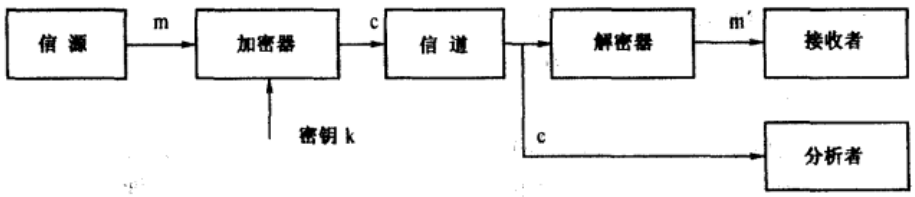


图 2-2　保密系统模型

保密系统设计的目的是使窃听者即使完全准确地接收到了传送消息也无法恢复出原始

消息。

1. 保密系统的数学模型

香农从概率统计观点出发研究信息的传输与保密。图 2-2 各部分可描述如下：

信源是产生消息的源，可以用简单概率空间描述离散无记忆源。设信源字母表为 $T=\{a_i, i=0,1,2,\cdots,N-1\}$，字母 a_i 出现的概率为 $p(a_i)\geqslant 0$，且 $\sum_{i=0}^{N-1}p(a_i)=1$。信源产生的任一长为 L 的消息序列为：

$$m=(m_1,m_2,\cdots,m_L),m_i\in T,1\leqslant i\leqslant L$$

若研究的是所有长为 L 的信源输出，则称

$$M=T^L=\{m=(m_1,m_2,\cdots,m_L)\mid,m_i\in T,1\leqslant i\leqslant L\}$$

为消息空间或明文空间，它含有 N^L 个元素。

密钥源是产生密钥序列的源。密钥通常是离散的，设密钥字母表为 $B=\{k_i, i=0,1,2,\cdots,s-1\}$。字母 k_i 的概率为 $p(k_i)\geqslant 0$，且 $\sum_{i=0}^{s-1}p(k_i)=1$。一般密钥源为无记忆均匀分布源，即各密钥符号独立等概。所有长为 r 的密钥序列 $k=k_1k_2\cdots k_r$，$k_i\in B$，$1\leqslant i\leqslant r$，称为密钥空间 K。一般的，消息空间与明文空间彼此独立。合法的接收者知道 k 和 K，而窃听者不知道 k，也无从确定 K。

加密变换是将明文中的元素 m 在密钥控制下变为密文 c 的过程，即 $c=(c_1,c_2,\cdots,c_L')=E_k(m_1,m_2,\cdots,m_L)$，称 c 的全体 $C=Y^{L'}$ 为密文空间，其中 Y 表示密文字母表。密文空间的统计特性由明文空间和密钥空间的统计特性完全决定。可证明如下：

设明文 $m\in M$ 发生的概率为 $P_M(m)$，密钥 k 被选择的概率为 $P_K(k)$，消息空间与密钥空间彼此独立。对密钥 $k\in K$，令 $C_k=\{E_k(m)\mid m\in M\}$。对于每一个 $c\in C$，有

$$p_C(c)=\sum_{|k|c\in C_k|}P_K(k)P_M(D_k(c)) \tag{2-1}$$

又 $P_C(c\mid m)=\sum_{|k|m=D_k(c)|}P_K(k)$，由贝叶斯公式可得

$$P_M(m\mid c)=\frac{P_M(m)\sum_{|k|m=D_k(c)|}P_K(k)}{\sum_{|k|c\in C_k|}P_K(k)P_M(D_k(c))} \tag{2-2}$$

由式(2-1)、式(2-2)可知，只要知道明文空间和密钥空间的概率分布就能确定出密文空间的概率分布和明文空间关于密文空间的条件概率分布。

一般假定保密系统的信道是无扰的，因而对于合法的接收者，他知道解密变换和密钥，能够很容易地从密文得到原来的消息。即

$$m=D_K(c)=D_K(E_K(m))$$

2. 熵与不确定性

熵是一种信息量的表示方法，即在等概率条件下对消息中所有可能的值进行编码所需要的最少位数，也可理解为表示不同消息值所需的平均比特数目。如“性别”信息有两种取值——“男”或“女”，可用 1 位表示，因此其熵值为 1。

定义 2-1 设 $X=\{x_i\mid i=1,2,\cdots,n\}$，$x_i$ 出现的概率为 $p(x_i)\geqslant 0$，且 $\sum_{i=1}^{n}p(x_i)=1$。集 X

的熵定义为：

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i) \tag{2-3}$$

其信息单位是比特(bit)。熵表示了消息的不确定性或为确定一个消息所需的信息量，即当消息被加密成密文时，为了获取明文，需要解密的明文的位数。例如密文“qwefg”可能是“男”，也可能是“女”，那么此消息的不确定性是 1，为恢复出该消息，密码分析者只需选择 1 位即可。熵也可理解为一条消息所给出的平均信息量。

【例 2-1】 令 $X=\{x_1, x_2, x_3\}$，$p(x_1)=1/2$，$p(x_2)=1/4$，$p(x_3)=1/4$，则 X 的熵为

$$H(X) = \left[-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{4}\log_2\frac{1}{4} = \frac{1}{2} + 2\times\left(\frac{1}{4}+\frac{1}{4}\right)\right]\text{bit} = 1.5\text{bit}$$

定义 2-2 设 $X=\{x_i \mid i=1, 2, \cdots, n\}$，$x_i$ 出现的概率为 $p(x_i)\geqslant 0$，且 $\sum_{i=1}^{n} p(x_i)=1$。$Y=\{y_j \mid j=1, 2, \cdots, m\}$，$y_j$ 出现的概率为 $p(y_j)\geqslant 0$，且 $\sum_{j=1}^{m} p(y_j)=1$，则联合事件集 $XY=\{x_iy_j \mid i=1, 2, \cdots, n, j=1, 2, \cdots, m\}$，令 x_iy_j 出现的概率为 $p(x_iy_j)\geqslant 0$，且 $\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_iy_j)=1$。集 X 和 Y 的联合熵定义为：

$$H(XY) = H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_iy_j)\log_2 p(x_iy_j) \tag{2-4}$$

集 X 相对于事件 $y_j \in Y$ 的条件熵为：

$$H(X \mid y_j) = -\sum_{i=1}^{n} p(x_i \mid y_j)\log_2 p(x_i \mid y_j) \tag{2-5}$$

集 X 相对于集 Y 的条件熵为：

$$H(X \mid Y) = \sum_{j=1}^{m} p(y_j)H(X \mid y_j) = -\sum_{j=1}^{m}\sum_{i=1}^{n} p(y_j)p(x_i \mid y_j)\log_2 p(x_i \mid y_j) \tag{2-6}$$

同样的，集 Y 相对于事件 $x_i \in X$ 的条件熵为：

$$H(Y \mid x_i) = -\sum_{j=1}^{m} p(y_j \mid x_i)\log_2 p(y_j \mid x_i) \tag{2-7}$$

集 Y 相对于集 X 的条件熵为：

$$H(Y \mid X) = \sum_{i=1}^{n} p(x_i)H(Y \mid x_i) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i)p(y_j \mid x_i)\log_2 p(y_j \mid x_i) \tag{2-8}$$

熵 $H(X|Y)$ 表示观察到集 Y 后集 X 还保留的不确定性，通常将它称为含糊度，将条件熵 $H(Y|X)$ 称为散布度。

联合事件的概率为 $p(x_iy_j)$，有：

$$\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_iy_j) = \sum_{i=1}^{n} p(x_i)\sum_{j=1}^{m} p(y_j \mid x_i) = \sum_{j=1}^{m} p(y_j)\sum_{i=1}^{n} p(x_i \mid y_j) = 1 \tag{2-9}$$

由以上各式，可以得到：

$$H(XY) = H(X) + H(Y \mid X) \tag{2-10}$$

$$= H(Y) + H(X \mid Y) \tag{2-11}$$

且可推得：

$$H(X \mid Y) \leqslant H(X) \tag{2-12}$$

$$H(Y \mid X) \leqslant H(Y) \tag{2-13}$$

$$H(X, Y) \leqslant H(X) + H(Y) \tag{2-14}$$

类似式2-4，对于 n 个事件集，定义 n 维联合熵

$$H(X_1X_2\cdots X_n) = H(X_1) + H(X_2 \mid X_1) + \cdots + H(X_n \mid X_1\cdots X_{n-1}) \tag{2-15}$$

$$\leqslant nH(X_1) \tag{2-16}$$

3. 完善保密性

完善保密性又称无条件保密性，指的是即使密码分析者具有无限资源也无法破译该系统。香农分析了惟密文攻击下保密系统的完善保密性。

由熵的概念，针对图2-2所示保密系统模型，令明文空间的熵为 $H(M)$，密钥空间的熵为 $H(K)$，密文空间的熵为 $H(C)$，在已知密文条件下明文与密钥的含糊度分别为 $H(M|C)$ 和 $H(K|C)$。当进行惟密文攻击时，密码分析者的任务就是从截获的密文中提取出有关明文的消息 $H(M) - H(M|C)$ 或者是有关密钥的消息 $H(K) - H(K|C)$，因此 $H(M|C)$ 和 $H(K|C)$ 越大，从密文中能够提取出的有关明文和密钥的信息量就越小。另外，

$$H(M \mid C) \leqslant H(M \mid C) + H(K \mid M, C)$$

$$= H(MK \mid C) = H(K \mid C) + H(M \mid CK)$$

因为在已知密文和密钥的前提下明文信息可全部确定，所以 $H(M|CK) = 0$。由上式可得 $H(M|C) \leqslant H(K|C) \leqslant H(K)$。这样

$$H(M) - H(M \mid C) \geqslant H(M) - H(K) \tag{2-17}$$

由式(2-17)可以看出，保密系统的密钥量越小，密文中含有的明文的信息量就越大。因此密码设计者应使密钥量足够大，一种极端情况就是使 $H(M) - H(M|C) = 0$。这正是前面所说的完善保密系统。

定义 2-3 如果 $H(M) = H(M|C)$，则这个保密系统是完善保密系统。

由式(2-17)可知：

定理 2-1 完善保密系统存在的必要条件是 $H(M) \leqslant H(K)$。

因此，要构造完善保密系统，其密钥量的对数（密钥空间均匀分布情况下）不能小于明文集的熵。当密钥为二元序列时要满足

$$H(M) \leqslant H(K) = H(k_1, k_2, \cdots, k_r) \leqslant r(\text{bit}) \tag{2-18}$$

利用构造法，可以证明：

定理 2-2 存在完善保密系统。

4. 惟一解距离与理论保密性

上面讨论的是在惟密文攻击下的系统安全性，Massey 也将香农提出的这一概念推广到已知明文攻击。本节将讨论在惟密文攻击下破译密码体制的理论问题。

在信息传输过程中，所传送的符号比实际信息所需要的符号多。这就为密码分析者的破译提供了可利用的条件，因此冗余度是密码分析的基础。冗余度越小，密码分析越困难，系统的保密性越好。香农定义惟一解距离来描述破译分析的困难性。惟一解距离实质就是能惟一确定密钥值的最少密文数量。

定义 2-4 对一种语言，其语言信息率为 $r = H(M)/N$，其中 N 是消息的长度。

当 N 足够大时，英语的语言信息率介于1位/字母与1.5位/字母之间。

假定每一个字符串是等可能的，每一个字母可被编码的最大位数称为该语言的绝对信息率。若某一语言有 L 个字母，其绝对信息率为 $R = \log_2 L$。

定义 2-5 一种语言的冗余度定义为 $D = R - r$。

比如，英语的绝对信息率为 $\log_2 26 = 4.7$ 位/字母，经随机估算得到的信息率为 1.3 位/字母，则其冗余度为 3.4 位/字母。

由于明文存在冗余度，实现完善保密所需的密钥量的上限就降低了，因此在加密前进行数据压缩是强化保密系统的重要措施。

假定密码分析者具有无限的计算资源，并且知道明文所使用的语言。当他截获到某一密文时，可以用所有可能的密钥解密，并记录下所有有意义的明文所对应的密钥，这样就构成了一个密钥集合，其元素既有正确的密钥，也有不正确的密钥。那些可能的但不正确的密钥称为伪密钥。以下定理反应了保密系统各部分之间的关系。

定理 2-3 对一保密系统，有 $H(K \mid C) = H(K) + H(M) - H(C)$

证明：由式(2-10)、式(2-11)，有

$$H(K,M,C) = H(C \mid K,M) + H(K,M) = H(M \mid K,C) + H(K,C)$$

由于密钥和明文惟一确定密文；密钥和密文惟一确定明文；密钥和明文统计独立，所以

$$H(C \mid K,M) = 0, H(M \mid K,C) = 0, H(K,M) = H(K) + H(M)$$

故 $$H(K) + H(M) = H(K,C) = H(C) + H(K \mid C)$$

所以 $$H(K \mid C) = H(K) + H(M) - H(C)$$

因此，随着截获密文的增多，$H(C)$ 增大，条件熵 $H(K \mid C)$ 必然变小，也就是说，能够得到密钥的不确定性也越来越小。

香农从给定 v 长密文序列集 $C = C_1, C_2, \cdots, C_v \in C^v$ 条件下密钥的不确定性出发，研究破译一种密码体制时密码分析者应处理的最少密文量，也就是从研究密钥含糊度 $H(K \mid C^v)$ 着手研究该问题。由条件熵性质可知

$$H(K \mid C_1, C_2, \cdots, C_{v+1}) \leqslant H(K \mid C_1, C_2, \cdots, C_v)$$

也就是说，随着 v 的增大，密钥含糊度非递增，即随着截获的密文增加，得到的有关明文或密钥的信息量就增加。当 $H(K \mid C^v) \to 0$ 时就可惟一地确定密钥 K，从而破译该密码。

定义 2-6 对于给定的密码系统，称

$$v_0 = \min\{v \in N \mid H(K \mid C^v) \approx 0\}, N \text{是整数} \tag{2-19}$$

为在惟密文攻击下的惟一解距离，一般用 u_d 表示。

当截获的密文量大于 v_0 时，理论上就可惟一地确定该系统的密钥。惟一解距离也可理解为使伪密钥的期望值等于 0 的 v 值。

可以证明：

定理 2-4

$$u_d = \frac{H(K)}{D} \tag{2-20}$$

综上所述，提高系统安全性的方法一种是增大 $H(K)$，即采用复杂的密码体制，直至一次一密密钥体制；另外就是减小冗余度。

定义 2-6 仅仅描述了理论上的可能性，假定分析者知道所用语言的全部统计特性，而实际上由于自然语言的复杂性，这是无法办到的；另外没有考虑为了得到惟一解所做的工作，没有考虑实际破译时的工作量。这也是将信息论用于密码学研究的一个缺陷，即只能分析密码体制的理论安全性。香农意识到了这一点，并提出了实际保密性的概念。

5. 实际保密性

理论保密性是假定密码分析者有无限的时间、设备、人力和资金前提下研究的密码系统的安全性。若在这种假定情况下，密码分析者截获了任意多密文仍无法破译该系统，则称它在理论上是保密的。而实际上密码分析者所具有的时间、设备等各种资源总是有限的，在这种条件下研究的密码系统的安全性就是其实际保密性。如果破译一个密码系统所需的工作超过了密码分析者的能力，则该系统就是实际安全的。

在考察一个系统的实际安全性时，主要考虑两个因素：一是密码分析者的计算能力；另外就是他所采用的破译算法的有效性。密码分析者的计算能力取决于他所拥有的设备条件，一般都假定他们拥有最好的设备。密码分析者总是寻找更为有效的破译算法以减少破译所需的工作量。

2.1.2 数学基础

现代密码学的发展离不开数学，本节将介绍一些基本的数学方面的知识，包括复杂性理论、数论及有限域上的离散对数问题等。其中，复杂性理论是密码分析的基础，结合复杂性理论，人们在对数论及离散对数问题研究的基础上发展了公钥密码体制，典型算法有 RSA 算法、ElGamal 算法等。随着密码技术的发展，越来越多的数学理论和数学难题被用于密码体制的设计与分析，将在后续各章中陆续进行介绍。

1. 复杂性理论

计算复杂性理论将需要用计算机处理的数学问题按困难性进行定量描述和分类，确定哪些问题可以在计算机上求解，求解时采用何种方法更好。因此，复杂性理论也提供了一种分析不同密码技术和算法的计算复杂性的方法，它对密码算法及技术进行比较，并确定它们的安全性。

定义 2-7 问题指要在计算机上求解的对象。问题的规模是求解问题时所需输入变量的个数或输入二元数据的位数。

定义 2-8 算法是求解某个问题的一系列基本步骤（程序）或方法。“一个算法能解一个问题”是指此算法可解该问题中的任一特例；“一个问题可解”是指至少有一种算法能对其进行求解；“最佳算法”是求解某个问题的最快或最节省资源的算法。

（1）算法复杂性

一个算法的复杂性是运行它所需的计算能力。算法复杂性常用概算法所需的计算时间（T）和存储空间（S）来度量，分别称为计算复杂性和空间复杂性。T 和 S 一般是输入 n 的函数 $f(n)$，通常 n 是输入数据的长度。n 越大，所需的时间代价和空间代价就越大。

算法复杂性 $f(n)$ 通常表示为 $O(g(n))$ 数量级形式，并称“$O(g(n))$”为“大 O”表示法。

定义 2-9 $f(n)=O(g(n))$ 指存在常数 c 和 n_0，使得对一切 $n\geqslant n_0$，有 $|f(n)|\leqslant c|g(n)|$。

例如，对 $f(n)=9n+2$，有 $f(n)=O(n)$，因为取 $n_0=2$，$g(n)=n$，$c=10$，当 $n\geqslant n_0$ 时，有 $|f(n)|\leqslant c|g(n)|$。

若 $f(n)$ 是多项式 $f(n)=a_t n^t+a_{t-1}n^{t-1}+\cdots+a_1 t+a_0$，$t$ 为常数，则 $f(n)=O(n^t)$，所有的低阶项和常数可忽略不计。

采用这种方法表示的计算复杂性与所用系统无关，不必知道每个指令的精确时间或用于表示不同数据的位数，也不必知道处理器的速度。同时，采用这种方法能够清楚地看出输入

长度是如何影响时间和空间需求的。例如，若 $T = O(n)$，则输入长度加倍，算法运行时间也加倍；若 $T = O(2^n)$，则输入长度增加一位，算法运行时间就加倍。

通常算法按其时间和空间复杂性进行分类。

定义 2-10 多项式时间算法是指时间复杂性为 $O(n^t)$ 的算法，其中 t 为常数，n 是输入数据的长度。若 $t=0$，则称它是常数级的，即算法复杂性不依赖于 n；若 $t=1$，则复杂性随 n 线性增长，即它是线性的；若 $t=2$，就称它是二次的，等等。

定义 2-11 指数时间算法是指时间复杂性为 $O(t^{h(n)})$ 的算法，其中 t 为常量，$h(n)$ 是多项式。当 $h(n)$ 大于常数而低于线性函数时，称为超多项式时间算法。

如时间复杂性为 $O(n^{\log_2 n})$，$O(e^{\sqrt{n\ln n}})$ 的算法都是超多项式时间算法。

当 n 增大时，算法的时间复杂性能够显示算法是否实际可行方面的巨大差异。密码设计者对密码算法的一个基本要求就是加/解密只需多项式时间代价，而任何攻击算法均具有指数级的时间复杂性。

（2）问题复杂性

复杂理论也对问题的内在复杂性进行分类。它按照问题最困难的特例求解时所需的最小时间或空间量级，对问题的复杂性进行分类，可以看作是在图灵机上解决最难的问题实例所需的最小时间和空间。

图灵机是一种具有无限长度的读写磁带的有限状态机，可分为确定型和非确定型两种。确定型图灵机是指图灵机的每一步操作结果是惟一确定的，非确定型图灵机是指图灵机的每一步操作结果及下一步操作都有多种选择，不是惟一确定的。

定义 2-12 在确定型图灵机上可在多项式时间内求解的问题称为 P 问题；在非确定型图灵机上能在多项式时间内求解的问题称为 NP 问题。

对于任何一个 NP 问题的实例，对所给出的任意一个猜测都可在多项式时间内得到验证。若所有 NP 问题都可以在多项式时间内利用确定型图灵机求解，则 NP = P，但至今没有证明是否 NP = P；另外，许多 NP 问题看上去比 P 问题困难得多，但至今也没能证明 NP ≠ P。旅行商问题、背包问题、整数分解问题等都属于 NP 问题。

若一个问题可在确定型图灵机上求解，则在算法的每一步计算中下一步是惟一确定的，称该算法是确定性算法。若一个问题是在非确定型图灵机上可解，则在算法的某个计算步骤中必须从一个有限的可选择项中选一个作为下一步的操作，该算法就是非确定性的。

定义 2-13 设 x_1 和 x_2 是两个问题，若 x_1 可用多项式时间的确定性算法转化为 x_2，而 x_2 的解又可用多项式时间的确定性算法转化为 x_1 的解，则称 x_1 可归约为 x_2，记作 $x_1 \propto x_2$。

利用归约的概念可将问题进行转化，若 $x_1 \propto x_2$，则对 x_1 的研究可转化为对 x_2 的研究。

在现代密码学中，一个密码系统的破译常常可归结为求解某个数学问题，因此计算复杂性理论为破译密码的复杂度提供了度量方法；而计算复杂性理论的一些数学问题又给人们提供了设计实用安全的密码系统的基础。因此，复杂性理论是现代密码体制设计与分析的重要基础。

2. 数论

数论是现代密码学中，尤其是公钥密码体制中最重要的数学基础，本节将给出一些必要的定义，并不加证明地列出对密码体制设计有用的数论理论。

（1）素数

素数是数论研究的重点。只能被 1 和它本身除尽的数称为素数，不是 1 且非素数的数称

为合数。素数是无限的，在密码学中，常用大素数。

定义 2-14 任意给定整数 a、b，$b\neq 0$，若存在一整数 q，使得 $a=bq$ 成立，就说 b 能整除 a，记作 $b\mid a$，换而言之，a 除以 b 的余数为 0。此时，把 b 叫做 a 的因数（因子），把 a 叫做 b 的倍数。如果不存在整数 q，使 $a=bq$ 成立，则 b 不能整除 a，记作 $b\nmid a$。

定义 2-15 对于 n 个不全为零的整数 a_1，a_2，…，a_n，若整数 d 是它们中每一个的因子，则称 d 是 a_1，a_2，…，a_n 的一个公因子。a_1，a_2，…，a_n 的公因子中最大的一个叫做最大公因子，记作 $(a_1, a_2, \cdots, a_n)$ 或 $\gcd(a_1, a_2, \cdots, a_n)$。若 $\gcd(a_1, a_2, \cdots, a_n)=1$，称 a_1，a_2，…，a_n 互素。

定理 2-5 （Wilson 定理）p 是素数的充分必要条件是

$$(p-1)!\equiv -1(\mathrm{mod}p) \tag{2-21}$$

（2）模运算

定义 2-16 令三整数 a、b 及 n，称 a 在模 n 时与 b 同余，当且仅当 a 与 b 的差为 n 的整数倍，即 $a-b=kn$，其中 k 为任一整数，记作 $a\equiv b\bmod n$。

由上述定义可知，若 a 在模 n 时与 b 同余，则用 n 分别去除 a 和 b 后所得的余数相同。同余具有以下性质：

自反性：$a\equiv a\bmod n$；

对称性：若 $a\equiv b\bmod n$，则 $b\equiv a\bmod n$；

传递性：若 $a\equiv b\bmod n$，且 $b\equiv c\bmod n$，则 $a\equiv c\bmod n$。

模运算同其他普通的运算一样是可交换、可结合、可分配的，而且还具有以下性质：

$(a+b)\bmod n=[(a\bmod n)+(b\bmod n)]\bmod n$

$(a-b)\bmod n=[(a\bmod n)-(b\bmod n)]\bmod n$

$(ab)\bmod n=[(a\bmod n)(b\bmod n)]\bmod n$

定理 2-6 若 $a\equiv b\bmod n$，$\alpha\equiv\beta\bmod n$，则有

1）$ax+\alpha y\equiv bx+\beta y\pmod n$

2）$a\alpha\equiv b\beta\pmod n$

3）$a^m\equiv b^m\pmod n$，$m>0$

（3）中国剩余定理

若整数 x_1 满足线性同余式 $ax\equiv b\bmod n$，即 $ax_1\equiv b\bmod n$，则所有模 m 与 x_1 同余的整数都满足该线性同余式。

定理 2-7 同余式 $ax\equiv b\bmod n$ 有解的充要条件是 $d\mid b$，其中 $d=\gcd(a, n)$。若 x_0 是该同余式的一个解，令 $m'=m/d$，则它的所有解 x 均满足 $x\equiv x_0\bmod m'$。

定理 2-8 下列两个同余式

$$x\equiv b_1\bmod n_1, x\equiv b_2\bmod n_2$$

有一个公共解的充要条件是 $b_1\equiv b_2\bmod d$，$d=\gcd(n_1, n_2)$。

定理 2-9 联立同余式 $x\equiv b_i\bmod n_i$，$i=1, 2, \cdots, n$，有一个公共解的充要条件是

$$\gcd(n_i, n_j)\mid(b_i-b_j), i\neq j, i, j=1, 2, \cdots, n$$

定理 2-10 （中国剩余定理）设 m_1，m_2，…，m_k 是两两互素的正整数，$M=m_1\cdots m_k$，$M_i=M/m_i$，$i=1$，…，k，则同余式组

$$x\equiv b_i\bmod m_i, i=1, 2, \cdots, k$$

有惟一解 $x \equiv b_1M_1y_1 + b_2M_2y_2 + \cdots + b_kM_ky_k \pmod{M}$，其中，$M_iy_i \equiv 1 \bmod m_i$，$i = 1, \cdots, k$。

【例 2-2】 求解

$$\begin{cases} x \equiv 0 \bmod 3 \\ x \equiv 1 \bmod 5 \\ x \equiv 2 \bmod 7 \end{cases}$$

解：$M = 105$，$M_1 = 35$，$M_2 = 21$，$M_3 = 15$，故

$$35y_1 \equiv 1 \bmod 3,\ y_1 = 2;$$

$$21y_2 \equiv 1 \bmod 5,\ y_2 = 1;$$

$$15y_3 \equiv 1 \bmod 7,\ y_3 = 1。$$

所以 $x = 35 \times 2 \times 0 + 21 \times 1 \times 1 + 15 \times 1 \times 2 \equiv 51 \bmod 105$

（4）欧拉函数

利用同余的概念，所有整数在模 n 中被分成 n 个不同的剩余类。例如所有模 n 和 r 同余的整数组成一个剩余类 C_r。一组整数 r_1，r_2，$\cdots$，r_n 分别属于模 n 的不同剩余类时，则称它构成一组模 n 的完全剩余集。显然，集合 $\{0, 1, 2, \cdots, n-1\}$ 是模 n 的一组完全剩余集，称为模 n 的非负最小完全剩余集，用 Z_n 表示。

在模 n 的完全剩余集中，将所有与 n 互素的剩余类组成一个集合，该集合称为模 n 的即约剩余集，用 Z_n^* 表示。例如 $n = 9$，$\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ 为模 9 的完全剩余集，而 $\{1, 2, 4, 5, 7, 8\}$ 为模 9 的即约剩余集。

定义 2-17 欧拉函数 $\varphi(n)$ 为小于 n 且与 n 互素的所有整数的个数，即 $\varphi(n)$ 为模 n 即约剩余集中所有元素的个数。

由定义知，若 p 是素数，则 $\varphi(p) = p - 1$。

定理 2-11 若 p 与 q 互素，则 $\varphi(pq) = \varphi(p)\varphi(q)$。

定理 2-12 （欧拉定理）若 $(a, n) = 1$，则 $a^{\varphi(n)} \equiv 1 \bmod n$。

由欧拉定理，可以推得费马定理：

定理 2-13 （费马定理）若 p 是素数，$(a, p) = 1$，则 $a^{p-1} \equiv 1 \bmod p$。

等价地，可以表示为：

定理 2-14 （费马定理）若 p 是素数，对任意正整数 a，有 $a^p \equiv a \bmod p$。

由以上可知，若 $(a, n) = 1$，则相对于模 n，a 的逆元素为 $a^{\varphi(n)-1}$。

【例 2-3】 若 $(a, n) = 1$，则 $ax \equiv b \bmod n$ 有惟一解 $x \equiv a^{\varphi(n)-1}b \pmod{n}$。

【例 2-4】 求 $7^{201} = x \bmod 11$。

解：$(7, 11) = 1$，所以 $7^{\varphi(11)} = 7^{10} = 1 \bmod 11$

$$7^{201} = (7^{10})^{20} \times 7 = 7 \bmod 11$$

（5）欧几里得算法

欧几里得算法通过简单的过程来确定两个正整数的最大公因子。如果这两个数互素，还能确定它们各自的逆元素。

定理 2-15 对任何非负整数 a、b，有 $\gcd(a, b) = \gcd(b, a \bmod b)$

可重复使用上式来求出最大公因子。

【例 2-5】 $\gcd(24, 18) = \gcd(18, 6) = \gcd(6, 0) = 6$

【例 2-6】 求 $\gcd(2067, 1971)$。

解：$2067 = 1 \times 1971 + 96$　　gcd（1971，96）

$1971 = 20 \times 96 + 51$　　gcd（96，51）

$96 = 1 \times 51 + 45$　　gcd（51，45）

$51 = 1 \times 45 + 6$　　gcd（45，6）

$45 = 7 \times 6 + 3$　　gcd（6，3）

$6 = 2 \times 3 + 0$　　gcd（3，0）

所以 gcd（2067，1971）$=3$。

定理 2-16　任意给定整数 $a>0$，$b>0$，存在两个整数 s、t，使得 $(a, b) = sa + tb$。

这样，当 a、b 互素时，存在两个常数 s、t 使得 $sa + tb = 1$。因此，$sa \equiv 1 \bmod b$，$tb \equiv 1 \bmod a$，所以 s 是 a 模 b 的逆元素，t 是 b 模 a 的逆元素。

【例 2-7】　求 W，使 $1001W \equiv 1 \bmod 3837$。

解：因为（1001，3837）$=1$，所以 W 存在且惟一。

又
$$3837 = 3 \times 1001 + 834$$
$$1001 = 1 \times 834 + 167$$
$$834 = 4 \times 167 + 166$$
$$167 = 166 + 1$$

所以 $1 = 167 - 166 = 167 - (834 - 4 \times 167) = 5 \times 167 - 834 = 5 \times (1001 - 834) - 834$

$= 5 \times 1001 - 6 \times 834 = 5 \times 1001 - 6 \times (3837 - 3 \times 1001) = 23 \times 1001 - 6 \times 3837$

故 $23 \times 1001 = 1 \bmod 3837$，所以 $W = 23 \bmod 3837$。

（6）平方剩余

定义 2-17　若 p 是奇素数，且 a 是一个整数，$1 \leqslant a \leqslant p-1$。若同余方程 $x^2 \equiv a \bmod p$，对一些 $x \in Z_p$ 成立，则称 a 是模 p 的平方剩余，也称为二次剩余。

例如，对于 $p=7$，因为 $1^2 = 1 \equiv 1 \bmod 7$，$2^2 = 4 \equiv 4 \bmod 7$，$3^2 = 9 \equiv 2 \bmod 7$，

$4^2 = 16 \equiv 2 \bmod 7$，$5^2 = 25 \equiv 4 \bmod 7$，$6^2 = 36 \equiv 1 \bmod 7$，所以其平方剩余是 1，2，4，每一个平方剩余均出现两次。

定理 2-17　若 p 是奇素数，则整数 1，2，…，$p-1$ 中正好有 $(p-1)/2$ 个是模 p 的平方剩余，其余的 $(p-1)/2$ 个是非平方剩余。

定义 2-18　设 p 是一个奇素数，对任何 $a \geqslant 0$，定义勒让德符号为 $L(a, p)$，

$$L(a,p) = \begin{cases} 0, & a \equiv 0 \bmod p \\ 1, & a \text{ 是模 } p \text{ 的平方剩余} \\ -1, & a \text{ 不是模 } p \text{ 的平方剩余} \end{cases}$$

定理 2-18　（欧拉准则）令 p 为一奇素数，则 $L(a, p) \equiv a^{(p-1)/2} \bmod p$。

定理 2-19　（整数的惟一分解定理）任一大于 1 的整数能表示成素数的乘积，即对于任一 $a>1$，有 $a = p_1 p_2 \cdots p_n$，$p_1 \leqslant p_2 \leqslant \cdots \leqslant p_n$，其中，$p_1$，$p_2$，…，$p_n$ 是素数。

由惟一分解定理可知，任一大于 1 的整数能够惟一地写成

$$a = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}, \alpha_i > 0 (i = 1, 2, \cdots, k)$$

其中，$p_i < p_j (i < j)$ 是素数。这也叫做 a 的标准分解式。

定义 2-19　假设 n 是一个奇正整数，n 的素数幂分解是 $p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$，设 $a \geqslant 0$ 是一个正整数，雅可比符号 $J(a, n)$ 定义为：

$$J(a,n) = \prod_{i=1}^{k} L(a,p_i)^{\alpha_i}$$

若 n 是素数，雅可比符号就是勒让德符号。

【例 2-8】 计算雅可比符号 $J(6278, 9975)$。

解：因为 $9975 = 3 \times 5^2 \times 7 \times 19$，所以

$$J(6278,9975) = L(6278,3) \times L(6278,5)^2 \times L(6278,7) \times L(6278,19)$$
$$= L(2,3) \times L(3,5)^2 \times L(6,7) \times L(8,19) = -1$$

3. 有限域上的离散对数

（1）群的基本概念

定义 2-20 一个集合及定义在这个集合上的一个或多个运算一起称为一个代数系统。

设集合 S 及其运算 $*$，$+$，…，用（S，$*$，$+$，…）表示一个代数系统。

定义 2-21 对一个代数系统（G，$*$），如果

1）运算“$*$”满足结合律，即对每个 a，b，$c \in G$，有 $(a*b)*c = a*(b*c)$；

2）存在单位元素 $e \in G$，即对每个 $a \in G$，有 $a*e = e*a = a$；

3）每个元素 $a \in G$，有其逆元素 $a^{-1} \in G$，即 $a*a^{-1} = a^{-1}*a = e$。

则称代数系统（G，$*$）为一个群，简写为 G 是一个群。

如果运算“$*$”还满足交换律，即对每个 a，$b \in G$，有 $a*b = b*a$，则群（G，$*$）为交换群，或阿贝尔群。

若群 G 的任意一个元素 b 均可表示成某一个固定元 a 的乘方，即在群 G 中，存在 $a \in G$，使得 $G = \{a^k \mid k \in Z\}$，则称群 G 为循环群。a 叫做群 G 的生成元，也说群 G 是由元素 a 生成的。

定义 2-22 群（G，$*$）中元素的数目称为群的阶，记作 $|G|$。如果 $|G|$ 有限，称（G，$*$）为有限群，否则为无限群。

定义 2-23 对于一个有限乘法群，定义一个元素 $g \in G$ 的阶为满足 $g^m = 1$ 的最小正整数 m 的值。

定理 2-20 如果 p 是一个素数，那么 Z_p^* 是一个循环群。

定义 2-24 设 p 为素数，$\alpha \in Z_p^*$ 是一个阶为 $p-1$ 的元素，则称 α 为模 p 的本原元，也称为生成元。

由此看出，α 是本原元当且仅当 $Z_p^* = \{\alpha^i \mid 0 \leqslant i \leqslant p-2\}$，即任意一个 $\beta \in Z_p^*$ 能惟一地写成 $\beta = \alpha^i$，$0 \leqslant i \leqslant p-2$。

定理 2-21 设 p 为素数，模 p 的本原元的数目是 $\varphi(p-1)$。

【例 2-9】 设 $p = 11$，则 2 是模 11 的本原元。这是因为

$2^1 = 2 \equiv 2 \bmod 11$；$2^2 = 4 \equiv 4 \bmod 11$；$2^3 = 8 \equiv 8 \bmod 11$；$2^4 = 16 \equiv 5 \bmod 11$；

$2^5 = 32 \equiv 10 \bmod 11$；$2^6 = 64 \equiv 9 \bmod 11$；$2^7 = 128 \equiv 7 \bmod 11$；

$2^8 = 256 \equiv 3 \bmod 11$；$2^9 = 512 \equiv 6 \bmod 11$；$2^{10} = 1024 \equiv 1 \bmod 11$。

同样可以验证模 11 的本原元还有 6、7、8，而其他的数都不是。

通常确定本原元是不容易的，但可以通过 $p-1$ 的素因子来进行验证。令 q_1，q_2，…，q_n 是 $p-1$ 的素因子，为测试数 g 是否为模 p 的本原元，对所有的 $q = q_1$，q_2，…，q_n，计算 $g^{(p-1)/q} \bmod p$，若对某个 q 值，其结果为 1，那么 g 不是一个本原元；若对任何 q 值，结果都

不等于1，那么 g 是本原元。

【例 2-10】 验证3是否为模11的本原元。

解：$11-1=10$ 的素因子是2和5，因为

$3^{10/5} \bmod 11 = 9$，$3^{10/2} \bmod 11 = 1$

所以3不是一个生成元。

（2）有限域

定义 2-24 若一个集合 F 在已定义的两个运算“+”、“*”中具有下列性质，则称 F 为一个域。

1）$(F, +)$ 为交换群，且具有单位元0；

2）$(F-\{0\}, *)$ 也为交换群；

3）F 中运算“*”对运算“+”满足分配律，即对于所有 a，b，$c \in F$，满足 $a*(b+c)=a*b+a*c$。

若域 F 的元素个数为无限，则 F 为无限域；反之，若其元素个数为有限，则 F 为有限域，记作 GF (p)，其中 p 表示域中元素的个数。

在有限域 F 中有一乘法单位元，以1表示，同时 F 在加法运算下也构成一个交换群，由有限域的有限性及封闭性可知，必存在一整数 n 使

$$\underbrace{1+1+\cdots+1}_{n}=0$$

使上式成立的最小整数 n 称为域 F 的特征。

定理 2-22 有限域的特征必为素数。

定理 2-23 $(F, +, *)$ 为一有限域 GF (p) 的充要条件是 p 是素数。

定义 2-25 令 $F_p[x]$ 为 GF (p) 上的多项式集合，x 的任意多项式可表示为

$$v(x)=v_0+v_1x+\cdots+v_{n-1}x^{n-1}=\sum_{i=0}^{n-1}v_ix^i \in F_p(x)$$

其中，$v_i \in$ GF (p)，$i=0, 1, \cdots, n-1$。

定义 $F_p[x]$ 上的多项式加法和多项式乘法，设两个多项式分别为

$$a(x)=\sum_{i=0}^{N-1}a_ix^i,\ b(x)=\sum_{i=0}^{M-1}b_ix^i$$

加法：$c(x)=a(x)+b(x)=\sum_{i=0}^{\max(N-1,M-1)}(a_i+b_i)x^i,\ c_i=a_i+b_i$

乘法：$c(x)=a(x)b(x)=\sum_{i=0}^{N+M-2}c_ix^i,\ c_i=\sum_{j=0}^{i}a_jb_{i-j}$

定义 2-26 若 $p(x) \in F[x]$，且除1以外所有次数低于 $p(x)$ 的多项式都除不尽 $p(x)$，则称 $p(x)$ 为即约多项式。

类似整数域，可以定义多项式的欧拉除法定理、惟一分解定理及剩余类等概念。

定义 2-27 令 $f(x)=f_0+f_1x+f_2x^2+\cdots+f_mx^m$ 为 GF(p) 上的 m 次多项式。令 E 为 GF(p) 上次数小于 m 的所有多项式，共有 p^m 个。定义模 $f(x)$ 的加法和乘法分别为：

模 $f(x)$ 加法 $\oplus$：$a(x) \oplus b(x)=R_{f(x)}[a(x)+b(x)]=a(x)+b(x)$

模 $f(x)$ 乘法 $\odot$：$a(x) \odot b(x)=R_{f(x)}[a(x)*b(x)]$

$R_{f(x)}M(x)$ 表示以 $f(x)$ 去除 $M(x)$ 所得的余式，也构成了模 $f(x)$ 下 $M(x)$ 的剩余类。

定理 2-24 $[E, \oplus, \odot]$ 为域 $GF(p^m)$ 的充要条件为 $f(x)$ 是 m 次即约多项式。

已经证明，有限域 F 只有两种，即 $GF(p)$ 和 $GF(p^m)$。

（3）有限域上的离散对数

模指数运算是密码学中的一种单向函数。设 p 是素数，$\alpha \in Z_p^*$ 是模 p 的本原元，在 Z_p 上已知 x，计算 $\alpha^x \bmod p$ 很容易；反之，若已知 $\beta \in Z_p^*$，求解 x 使得 $\alpha^x = \beta \bmod p$ 则很困难。x 可以表示为 $\log_\alpha \beta$，也称为求解离散对数。并不是所有的离散对数都有解。

有些密码体制的设计就是利用了求解离散对数的困难性。一般经常使用的有限域上的离散对数为：

1）素数域的乘法群：$GF(p)$；

2）特征为 2 的有限域上的乘法群：$GF(2^n)$；

3）有限域 F 上的椭圆曲线群：$EC(F)$。

2.2 现代密码技术分类

由第 1 章知道，一个密码系统主要由明文空间、密文空间、密钥空间、加密算法、解密算法 5 部分组成，其基本单元是算法和密钥。根据对密码系统中各部分处理方式的不同，可从不同的角度对现代密码技术进行分类。

2.2.1 流密码与分组密码

根据对明文的划分可将密码算法分为流密码和分组密码。分组密码将明文分为多组明文块（每块含有多个字符），逐组地进行加密，并且每一组的加密密钥相同；而在流密码中则是将明文消息按字符或位逐个加密，流密码也称为序列密码。

1. 分组密码

（1）分组密码概述

分组密码将明文 M 划分为固定长度的明文块 $M_1, M_2, \cdots, M_n$，对每一块都用相同的密钥 K 进行加密，即 $C = (C_1, C_2, \cdots, C_n)$，其中 $C_i = E(M_i, K), i = 1, 2, \cdots, n$。

分组密码实际上可以看作一类置换，算法设计要求可归纳为分组长度及密钥量都要足够大，加密算法要足够复杂，同时加密和解密运算简单，便于软件和硬件实现。

为实现复杂的加密算法，人们一般都遵循香农提出的混淆原则和扩散原则。所谓混淆是指密钥、明文及密文之间的依赖关系足够复杂，使得密码分析者无法利用它们之间的关系；扩散指密钥的每一位数字能影响密文的许多位，从而防止对密钥进行逐段破译，同时明文的每一位数字也能够影响许多位密文，以隐蔽明文的数字统计特性。

为了便于算法的软件实现，都使用子块和简单的运算，而且子块的长度能适应软件编程，如将子块确定为 8bit、16bit 或 32bit。针对硬件实现，都尽量使用规则结构，且使加密和解密可用同样的器件实现。

（2）分组密码的运行模式

在实际使用时，针对不同的性能要求和运行环境，分组密码可以采用不同的工作模式。

1）电子密码本模式（Electronic Code Book，ECB）。

电子密码本模式是使用分组密码的最简单的做法，即每个明文组独立地用同一密钥加密

成一个密文分组。由于加密是独立进行的，因此可以不按分组的顺序进行，这对于加密随机存取的文件是很合适的。如对于数据库，若采用 ECB 模式加密，任一个记录都能独立于其他的记录被加密、删除、添加等。

同时，采用 ECB 模式，相同的明文总会加密成相同的密文，理论上可以制造出一个包含明文和相应密文的密码本。这样很容易暴露明文数据的格式和统计特征。如果密码分析者有很多消息的明、密文，那么他也可以在不知道密钥的情况下编辑密码本。因此，采用 ECB 模式，密码分析者很容易实行统计分析攻击、分组重放攻击及代换攻击等。

上述缺陷产生的根本原因在于明文分组是被独立处理的，密码分析者可以按组进行分析。为此，人们提出了各种链接、反馈等模式来克服这一弱点。

2）密码分组链接模式（Cipher Block Chaining，CBC）。

链接模式是将一种反馈机制加入到分组密码中，也就是说，前一个分组的加密结果被反馈到当前分组的加密中。由于每一个分组都用来修改下一个分组的加密，所以每个密文分组不仅依赖于产生它的明文分组，还依赖于它前面的所有分组。密码分组链接模式的加/解密分别如图 2-3、图 2-4 所示。

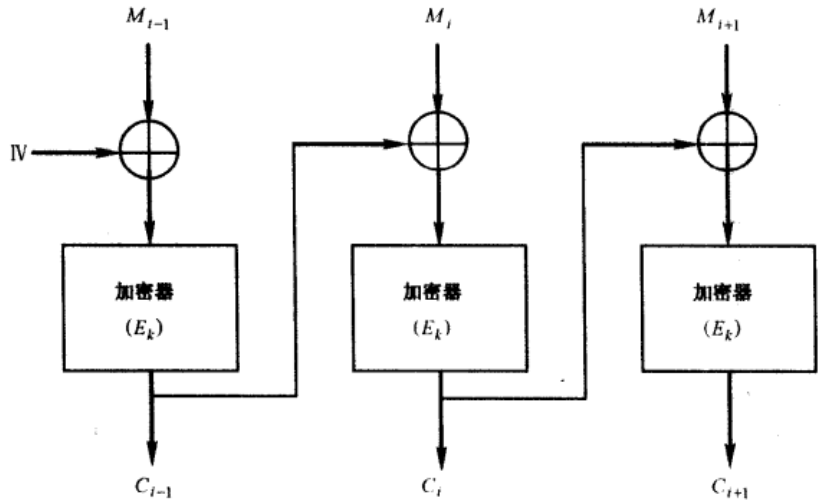


图 2-3　CBC 模式加密

由两图可以看到，在 CBC 模式中，第一个分组明文被加密后，其结果被存在反馈寄存器中，并与下一组明文相异或，作为下一次加密的输入；解密操作是上述加密操作的逆过程，首先第一组密文正常解密，同时将该密文存入反馈寄存器中，下一组密文正常解密后，其结果将与反馈寄存器中的结果进行异或操作，作为该组解密的明文结果。

在上述两图中的 IV 称为初始化向量。加入初始化向量有两个目的，第一是由于加密时第一组明文还没有反馈密文，可以为它预设一个初始化反馈向量。这就要求收发双方的 IV 相同，以便正确地进行加/解密；第二是由于在 CBC 模式中，仅在前面的明文分组不同时，才能将完全相同的明文分组加密成不同的密文分组，因此任意两则消息在它们第一次出现不同之前都将加密成同样的形式，这就使密码分析者有机可乘。一个有效的解决办法就是加密随机数据作为第一个分组，也就是随机选择一个初始化向量。使用IV后，相同的明文可以

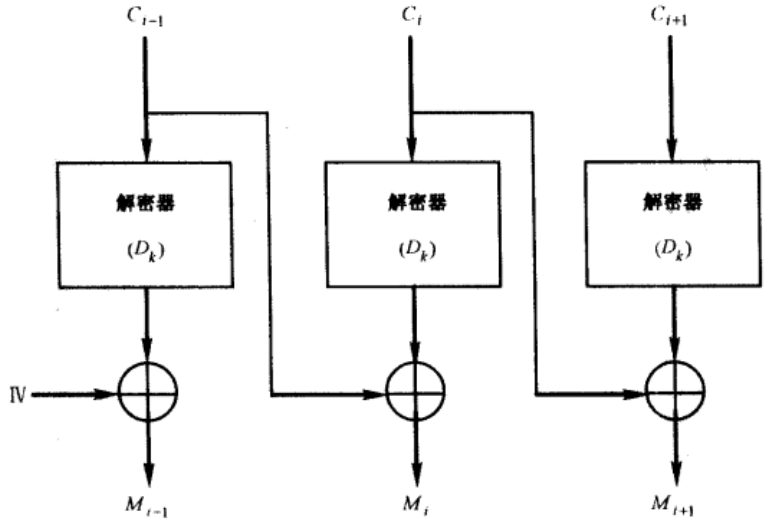


图 2-4　CBC 模式解密

被加密成不同的密文。IV 不需要保密，它可以与密文一道，以明文的形式发送给接收者；同时 IV 需要随时更新，以增加密码分析的困难性。

对 CBC 模式讨论较多的还有一个错误扩散问题。由于加密时的密文反馈性质，使得密文中任何一位发生变化都会影响后续的所有密文分组。这有两种情况，一种是一组明文发生错误，这会使得该组的密文及以后的所有密文都出错，但解密后，恢复的明文仍是这个错误，也就是说，除原来就有错的明文外，其余各组都能正确地恢复出明文。另外一种情况则是在传输过程中，由于信道噪音或介质损坏导致某一组密文出错，这不仅会使该组的明文恢复错误，同时由于解密的密文前馈性质，也会导致下一分组的密文解密错误，但后续分组不会再受错误影响。因此 CBC 模式的错误传播仅为 2 组，它具有自恢复能力。不过该结论仅针对自同步方式下的分组密码体制。如果出现了同步错误，例如在密文序列中增加或丢失了一位，将导致整个解密错误，所以采用 CBC 模式必须确保系统的帧同步。

3）密码反馈模式（Cipher Feedback，CFB）。

在 CBC 模式下，一个数据分组必须在接收完后才能进行加/解密运算，而许多网络应用要求在比分组小的单元里加密，比如电传、电报等，消息是按字符或比特处理的，这时可以采用密码反馈模式进行处理。

假设在 N 位分组密码算法下，采用 q 位密码反馈，如图 2-5、图 2-6 所示。操作将在 N 位队列中进行，每个队列单元为 q 位，图中的 M_i 及 C_i 均为 q 比特，$L = [N/q]$。最初，与 CBC 模式一样，先随机选择一初始化向量Ⅳ填入队列中，并对该队列进行加密。加密后最左端的 q 位密文与最初的 q 位明文异或，生成最初的 q 位密文。然后该密文还被反馈至队列的最右端，队列中的其他内容依次左移，抛弃最左边的单元内容。其余的明文依次操作。解密是该过程的逆过程。密码反馈模式下的加/解密操作均采用分组算法的加密函数。

由此可以看出，CFB 反馈的密文只是 q 位，而且它不直接与明文异或，而是反馈至密文队列。后面可以看到，这实际上是分组密码用于自同步流密码的一种情况。采用 CFB 模式，可以适应用户对数据格式的不同要求。

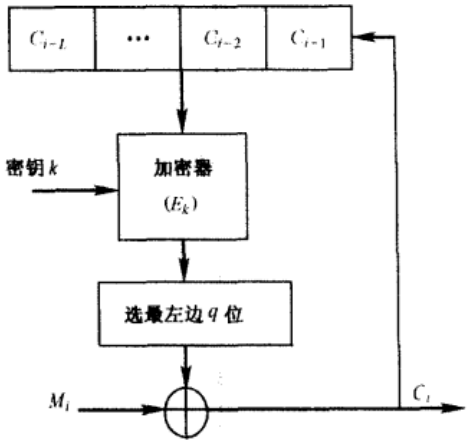


图 2-5　CFB 模式加密

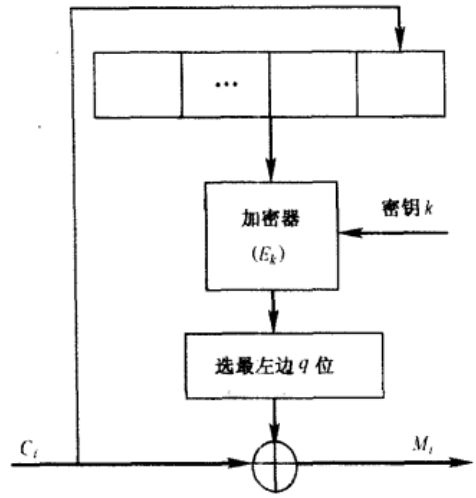


图 2-6　CFB 模式解密

在 CFB 模式中也有初始化向量的问题。在一次消息加密过程中，该Ⅳ必须惟一，否则密码分析者就可以分析出明文，而 CBC 没有这一要求；而且对不同的消息，Ⅳ必须更换。

同样，CFB 模式也存在错误扩散现象。明文的一个错误就会影响所有后面的密文及相应的解密结果；而密文的 1 位错误将导致解密明文的一个单独的错误。以 8 位 CFB 模式为例，在解密端由于前馈作用和移位存储器的作用，将使得连续 9 组的解密结果错误。一般而言，在 q 位的 CFB 模式中，一个密文错误会影响当前及随后的 $N/q-1$ 组的解密。对于同步错误来说，CFB 模式也是自恢复的。

4）输出反馈模式（Output Feedback，OFB）。

输出反馈模式与密码反馈模式类似，不同的是，它将加密器输出的最左边 q 位直接反馈至队列最右端的位置，同时这 q 位密钥与输入的 q 位明文异或，生成密文。这是为了克服 CBC、CFB 模式的错误扩散缺陷而提出来的，如图 2-7、图 2-8 所示。

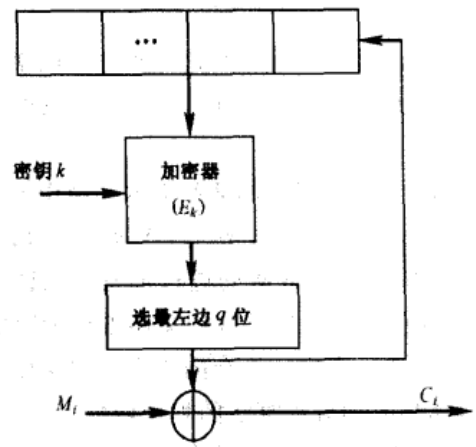


图 2-7　OFB 模式加密

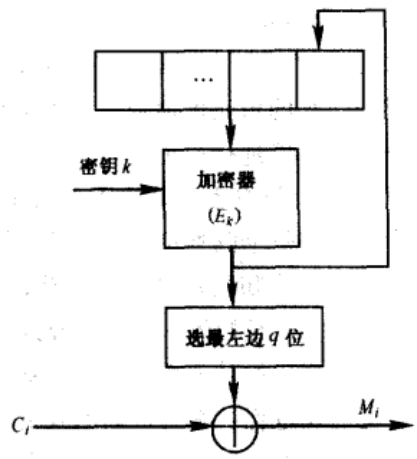


图 2-8　OFB 模式解密

与 CFB 模式一样，OFB 模式下的加/解密操作都使用分组算法的加密函数。OFB 模式最初也要使用初始化向量，并且应当惟一。对于 N 位输出反馈的 N 位分组密码来说，OFB 模式可以表示为

$$C_i = M_i \oplus S_i, S_i = E_k(S_{i-1})$$

$$M_i = C_i \oplus S_i, S_i = E_k(S_{i-1})$$

可以看到，该反馈机制独立于明文、密文，也称为内反馈机制。OFB 模式不存在错误扩散现象，但不具有自同步功能，如果加/解密两端的移位寄存器不同，将导致解密失败。因此实际系统必须有检测不同步并选取新的Ⅳ填充双方的移位队列以重新获得同步的机制。由后面内容可以知道，OFB 模式是将分组密码作为自同步序列密码运行的一种机制。

上述的4种模式具有不同的特点，适用于不同的环境。其中，ECB 是最快、最简单的分组密码模式，但它的安全性也最弱，一般不推荐使用 ECB 加密消息，但如果是加密随机数据，如密钥，ECB 则是很好的选择。CBC 适用于文件加密，而且有少量错误时不会造成同步失败，它是软件加密的最好选择。相比之下，OFB 和 CFB 要比 CBC 慢很多，CFB 通常是加密字符序列所选择的模式，它也能容忍少量错误扩展，且具有同步恢复功能。在极易出错的环境中可以选用 OFB 模式，但需有高速同步机制。

在实际应用中，上述四种模式几乎可以满足任何需要，它们既不会降低系统的安全性，也不是特别复杂。除此之外还有一些别的加密模式，如计数器模式、分组链接模式、明文分组链接模式、明文反馈模式等，实际应用不是很广泛，在此不再详述。

2. 流密码

（1）流密码的基本概念

流密码将明文 M 划分为一系列的字符或位 m_1，m_2，…，m_n，对每一个 m_i 用密钥序列 $K=(k_1, k_2, \cdots, k_n)$ 的第 i 个分量 k_i 进行加密，即 $C=(c_1, c_2, \cdots, c_n)$，其中，$c_i = E(m_i, k_i)$，$i=1, 2, \cdots, n$。一般情况下，加密算法都采用异或运算。解密时采用同步产生的同样的密钥流实现。流密码原理如图 2-9 所示。

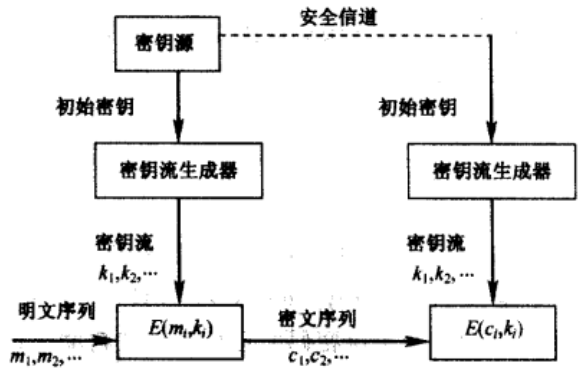


图 2-9　流密码原理图

由此可见，流密码的强度完全依赖于所产生的密钥流的随机性和不可预测性，流密码的核心问题就是密钥流生成器的设计算法。如若密钥流生成器的输出为 0 序列，则在异或运算下密文就是明文，该加密毫无价值；若输出是一无限随机序列，则该流密码就具有“一次一密”加密体制的完善保密性。一般来说，密钥流都是一个由初始密钥 k_1 通过确定性算法产生的伪随机序列，因此系统不再是完全保密的。若存在某个固定的整数 r，使得一个流密码的密钥流每隔 r 个字符（或位）后就重复，则称为周期序列流密码，反之，若密钥流永不重复，就是非周期序列流密码。在设计密钥流生成器时，都希望它产生的伪随机序列具有极大的周期、良好的随机统计特性及充分大的线性不可预测性。

同时，在流密码中要实现可靠解密还必须依赖于收发双方密钥流的精确同步。这里从密钥流生成器的基本结构谈起。如图 2-10 所示，密钥流生成器有三个基本的组成部分：内部状态、下一状态函数和输出函数。

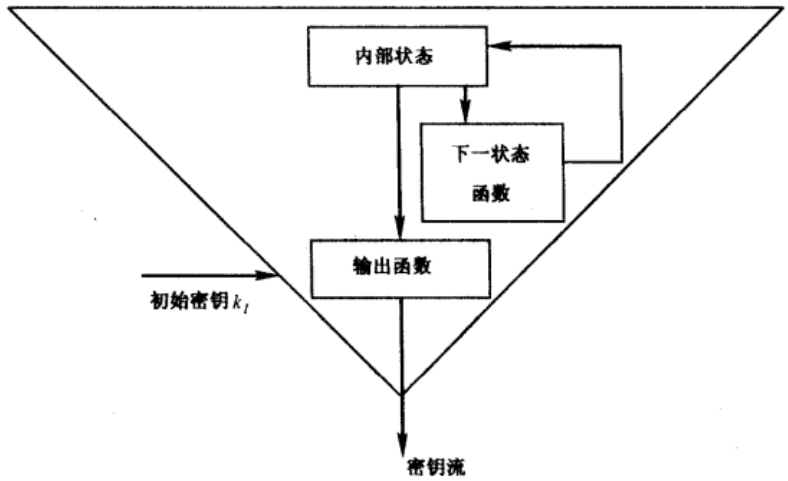


图 2-10　密钥流生成器内部原理

其中，内部状态表示密钥流生成器的当前状态，输出函数处理内部状态产生密钥序列，下一状态函数处理内部状态产生新的内部状态。可用有限状态自动机来描述它们之间的关系，如图 2-11 所示。

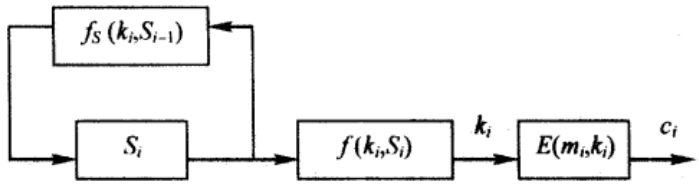


图 2-11　密钥流生成器有限状态自动机描述

图中 f（）为密钥流输出函数，f_S（）为下一状态函数，k_1 表示初始密钥。

$c_i = E\ (m_i,\ k_i)$

$m_i = D\ (c_i,\ k_i)$

$k_i = f\ (k_1,\ S_i)$

$S_i = f_S\ (k_1,\ S_{i-1})$

若 S_i 与明文消息无关，即密钥流独立于明文序列，称这类流密码为同步流密码，也称为密钥自动密钥（Key Auto-Key，KAK）；若密钥序列的每一位都是前面固定数量密文位的函数，则为自同步流密码，也称为密文自动密钥（Cipher Text Auto Key，CTAK）。

在同步流密码中，只要收发两端的密钥流生成器的初始密钥和初始状态相同，它们所产生的密钥就相同，但必须保持双方精确同步，如果传输过程中丢失了一位密文位，则之后的每一个密文都不能正确解密，必须重新进行同步。这样系统对插入、删除、重放等主动攻击很敏感，从而有利于检测系统所受到的安全攻击；同时该密码体制不存在差错传播问题，一些偶然错误只影响相应位的解密结果。相对于明文来说，同步流密码是无记忆的。

对于自同步流密码而言，由于密钥流与前面固定 n 位的密文有关，内部状态也依赖于前

面 n 位密文，如图 2-12 所示，所以解密密钥流生成器在连续收到 n 位密文后将自动与加密密钥流生成器同步。这一方面使得它对一些主动攻击不是很敏感，另一方面由于它将每个明文字符扩散到多个密文字符中，从而加强了它抗统计分析的能力。不过自同步流密码存在错误扩散现象，例如一位密文传输错误，将导致解密密钥流生成器后续 n 位密钥错误，从而导致相应的 n 位解密明文错误，之后两个密钥流生成器重新同步。

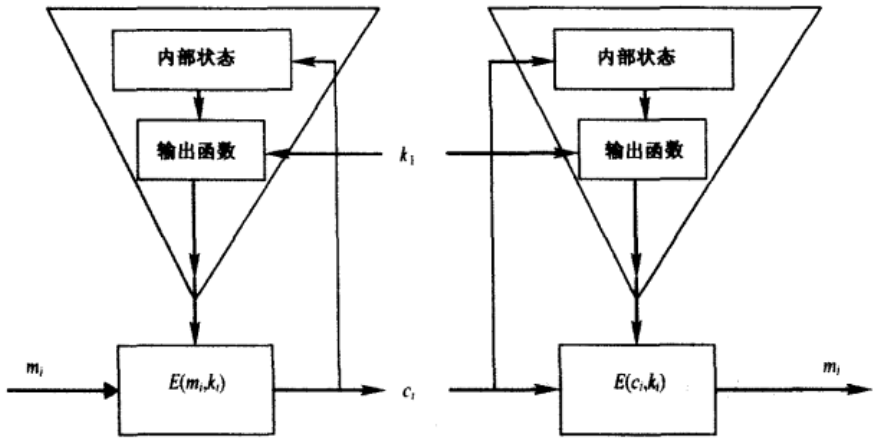


图 2-12　自同步密钥流生成器

如前所述，流密码中密钥流生成器的设计与密码系统的安全性有很大的关系。我们希望密钥流的产生能够保证系统的安全性，同时密钥流的产生要简单、快速，产生的密钥易于分配、保管、更换。移位寄存器是产生密钥序列的一个主要部分。一个反馈移位寄存器如图 2-13 所示。

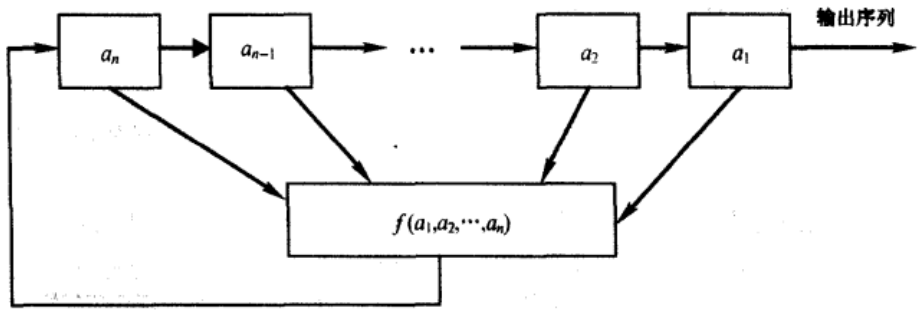


图 2-13　反馈移位寄存器

图中存储单元的个数 n 称为反馈移位寄存器的级数，在某一时刻，n 个存储单元的内容构成的向量（a_1，a_2，…，a_n）称为该移位寄存器的状态，函数 f（a_1，a_2，…，a_n）为其反馈函数，当反馈函数为线性函数时，称为线性反馈移位寄存器（LFSR），反之称为非线性反馈移位寄存器（NLFSR）。一般考虑二值序列，即每个寄存器单元的值非 0 即 1，那么在有限域 GF（2）中共有 2^{2^n}种反馈函数。输出序列的确定如下：设在 t 时刻反馈移位寄存器的状态为 S_t =（a_t，a_{t+1}，…，a_{t+n-1}），首先将移位寄存器的各个数据依次向前推进一位作为相应寄存器单元下一时刻的内容，同时整个移位寄存器的输出为 a_t，然后根据反馈函数计算 f（a_t，a_{t-1}，…，a_{t+n-1}）作为 a_{t+n-1}下一时刻的值，即 a_{t+n} =f（a_t，a_{t-1}，…，a_{t+n-1}），这样在下一时刻寄存器的状态为 S_{t+1} =（a_{t+1}，a_{t+2}，…，a_{t+n}）。随着时间的推移，可得到

输出序列 a_1，a_2，…，a_n，…，也称为反馈移位寄存器序列。

【例 2-11】 一个四级移位寄存器的初始状态为（a_4，a_3，a_2，a_1）=（1，0，1，1），反馈函数为 $f(a_1, a_2, a_3, a_4) = a_1 \oplus a_4 \oplus a_2 a_3$，则该移位寄存器的输出序列为：

a_4	a_3	a_2	a_1	输出
1	1	0	1	1
0	1	1	0	0
1	0	1	1	1
0	1	0	1	1
1	0	1	0	0
1	1	0	1	1
0	1	1	0	0
1	0	1	1	1
0	1	0	1	1
1	0	1	0	0
…	…	…	…	…

可见其输出序列为 1011010110…。

定义 2-28 若存在正整数 K，使得序列 a_1，a_2，…，a_n，…有

$$a_{i+K} = a_i, i = 1, 2, \cdots \tag{2-22}$$

则称序列 a_1，a_2，…，a_n，…为周期序列，满足式（2-22）的最小正整数 K 为该序列的周期。

例 2-11 中输出序列的周期为 5。

下面将分别介绍 GF（2）上的线性序列与非线性序列，以揭示流密码的基本原理。

（2）线性反馈移位寄存器

由于线性移位寄存器的反馈函数 $f(a_1, a_2, \cdots, a_n)$ 是 a_1，a_2，…，a_n 的线性函数，因此函数 f（）可表示为：

$$f(a_1, a_2, \cdots, a_n) = c_n a_1 \oplus c_{n-1} a_2 \oplus \cdots \oplus c_1 a_n \tag{2-23}$$

其中，c_i（$i=1$，2，…，n）为反馈系数，在二进制下可取 0 或 1。这样的线性函数共有 2^n 个。

如果以断开或闭合来分别表示 0 或 1，则线性移位寄存器可如图 2-14 表示。

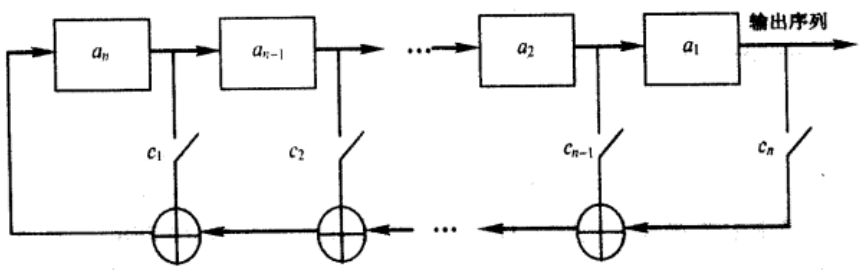


图 2-14 线性移位寄存器

对于 n 级线性移位寄存器最多有 2^n 个状态，而全 0 状态不会转入其他状态，因此线性移位寄存器的最大周期为 $2^n - 1$，输出序列的周期与状态周期相同，也小于或等于 $2^n - 1$。将这些非 0 序列的全体记作 $\Omega(f(x))$。

定义 2-29 周期为 $2^n - 1$ 的 LFSR 序列称为 m 序列。

以下讨论 m 序列的产生条件。

定义 2-30 以线性反馈移位寄存器的反馈系数决定的多项式

$$f(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1} + c_n x^n \tag{2-24}$$

称作 LFSR 的特征多项式或联系多项式。其中，$c_0 = c_n = 1$。

那么 $\Omega(f(x))$ 是特征多项式为 $f(x)$ 的 LFSR 的所有输出序列集。

定义 2-31 给定序列 $\{a_j\}$，$j \geq 1$，可以构成下述幂级数的形式：

$$A(x) = \sum_{i=1}^{\infty} a_i x^{i-1}$$

将其称为序列的生成函数，也称为母函数。

定理 2-25 $A(x) = \dfrac{\phi(x)}{f(x)}$，其中，$\phi(x) = \sum_{i=1}^{n} c_{n-i} x^{n-i} \sum_{j=1}^{i} a_j x^{j-1}$。

定理 2-26 $f(x) \mid p(x)$ 的充要条件是 $\Omega(f(x)) \subset \Omega(p(x))$。

该定理表明能用 n 级线性移位寄存器产生的序列，也可以用级数更多的线性移位寄存器产生。

定义 2-32 设 $f(x)$ 为 GF（2）上的多项式，使 $f(x) \mid x^{n-1}$ 的最小 n 称为 $f(x)$ 的周期。

定理 2-27 设 $f(x)$ 为 GF（2）上的 n 次多项式，且序列 $\{a_i\} \in \Omega(f(x))$，若 $f(x)$ 的周期为 p，则序列 $\{a_i\}$ 的周期 $p' \mid p$。

定理 2-28 设 $f(x)$ 是周期为 p 的 n 次即约多项式，$\{a_i\} \in \Omega(f(x))$，则序列 $\{a_i\}$ 的周期为 p。

定义 2-33 设 $f(x)$ 是 n 次即约多项式，若其周期为 $2^n - 1$，则称 $f(x)$ 是 n 次本原多项式。

定理 2-29 以 $f(x)$ 为特征多项式的 LFSR 的输出序列是 m 序列的充要条件是 $f(x)$ 为本原多项式。

下面从讨论随机序列的一般特性入手来讨论 m 序列的伪随机性。

定义 2-34 对于序列 $\{a_i\}$，若 $a_{t-1} \neq a_t = a_{t+1} = \cdots = a_{t+k-1} \neq a_{t+k}$，则称 $(a_t, a_{t+1}, \cdots, a_{t+k-1})$ 为一个长为 k 的游程。对于 0－1 序列，可将其称为 0 游程或 1 游程。

定义 2-35 GF（2）上周期为 T 的序列 $\{a_i\}$ 的自相关函数为：

$$R_a(\tau) = \frac{1}{T} \sum_{k=1}^{T} (-1)^{a_k} (-1)^{a_{k+\tau}}, 0 \leqslant \tau \leqslant T - 1$$

自相关函数实际上是序列 $\{a_i\}$ 与 $\{a_{i+\tau}\}$ 在一个周期内对应位相同的位数与对应位相异的位数之差的一个函数，即用相同位的位数减去不同位的位数，差值再除以周期 T。当 $\tau = 0$ 时，$R_a(\tau) = 1$；当 $\tau \neq 0$ 时，称 $R_a(\tau)$ 为异相自相关函数。由此可见，异相自相关函数是序列随机性的一个指标。

定理 2-30 GF（2）上的 n 级 m 序列 $\{a_i\}$ 具有如下性质：

1）在一个周期内，0、1 出现的次数分别为 $2^{n-1} - 1$ 和 2^{n-1} 次。

2）在一个周期内，总游程数为 2^{n-1}；对于 $1 \leqslant i \leqslant n-2$，长为 i 的游程有 2^{n-i-1}，且 0、1 游程各一半；长为 $n-1$ 的 0 游程 1 个；长为 n 的 1 游程 1 个。

3）$\{a_i\}$ 的自相关函数为：

$$R_a(\tau) = \begin{cases} 1, \tau = 0 \\ -\dfrac{1}{2^n - 1}, \tau \neq 0, 0 < \tau \leqslant 2^n - 2 \end{cases}$$

即异相关函数是一个常数。

由此可见 m 序列具有很好的随机统计特性。

综上所述，特征多项式 $f(x)$ 完全决定了线性移位寄存器输出序列的性质，当该多项式是本原多项式时，输出序列为 m 序列，该序列的周期最长（2^{n-1}），且具有良好的随机性。因此可以考虑采用线性反馈移位寄存器产生的 m 序列作为密钥流，进而设计加密算法，如图 2-15 所示。

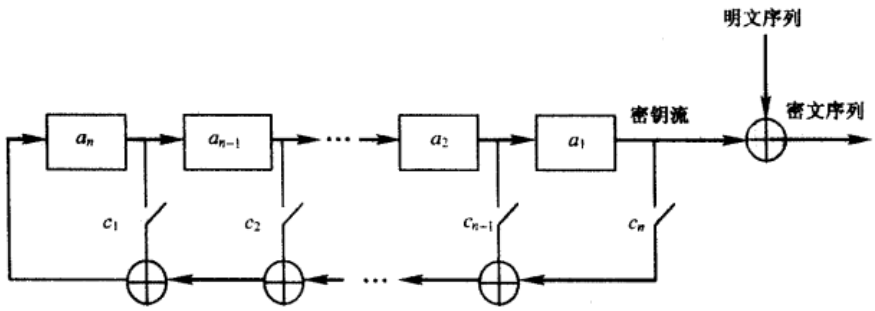


图 2-15　采用线性反馈移位寄存器设计加密算法

下面考察该加密方案的安全性。

设 $\boldsymbol{S}_h$ 和 $\boldsymbol{S}_{h+1}$ 分别表示线性移位寄存器的任两个连续状态，即输出序列向量，其中

$$\boldsymbol{S}_h = \begin{pmatrix} a_h \\ a_{h+1} \\ \vdots \\ a_{h+n-1} \end{pmatrix}, \boldsymbol{S}_{h+1} = \begin{pmatrix} a_{h+1} \\ a_{h+2} \\ \vdots \\ a_{h+n} \end{pmatrix}$$

同时，序列 $\{a_i\}$ 满足线性关系 $a_{h+n} = c_1 a_{h+n-1} \oplus c_2 a_{h+n-2} \oplus \cdots \oplus c_n a_h$，表示成矩阵形式为：

$$\begin{pmatrix} a_{h+1} \\ a_{h+2} \\ \vdots \\ a_{h+n} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ c_n & c_{n-1} & c_{n-2} & \cdots & c_1 \end{pmatrix} \begin{pmatrix} a_h \\ a_{h+1} \\ \vdots \\ a_{h+n-2} \\ a_{h+n-1} \end{pmatrix}$$

即 $\boldsymbol{S}_{h+1} = \boldsymbol{M}\boldsymbol{S}_h$，　　$\boldsymbol{M} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ c_n & c_{n-1} & c_{n-2} & \cdots & c_1 \end{pmatrix}$

该矩阵也称为反馈多项式 $f(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1} + c_n x^n$ 的伴侣矩阵。

在已知明文攻击下，假设破译者已经知道了 $2n$ 位明密文对 $\boldsymbol{M} = (m_1, m_2, \cdots, m_{2n})$，

$C=(c_1, c_2, \cdots, c_{2n})$，则可确定一段 $2n$ 位长的密钥序列 $K=(k_1, k_2, \cdots, k_{2n})$，$k_i=m_i \oplus c_i=m_i \oplus (m_i \oplus c_i)$，由此可确定该线性反馈移位寄存器的连续 $n+1$ 个状态：

$$\boldsymbol{S}_1=\begin{pmatrix}k_1\\k_2\\\vdots\\k_n\end{pmatrix}=\begin{pmatrix}a_1\\a_2\\\vdots\\a_n\end{pmatrix}, \boldsymbol{S}_2=\begin{pmatrix}k_2\\k_3\\\vdots\\k_{n+1}\end{pmatrix}=\begin{pmatrix}a_2\\a_3\\\vdots\\a_{n+1}\end{pmatrix}, \cdots, \boldsymbol{S}_{n+1}=\begin{pmatrix}k_{n+1}\\k_{n+2}\\\vdots\\k_{2n}\end{pmatrix}=\begin{pmatrix}a_{n+1}\\a_{n+2}\\\vdots\\a_{2n}\end{pmatrix}$$

组成矩阵 $\boldsymbol{X}=[\boldsymbol{S}_1, \boldsymbol{S}_2, \cdots \boldsymbol{S}_n]$，同时

$$[a_{n+1}\ a_{n+2}\ \cdots\ a_{2n}]=[c_n\ c_{n-1}\ \cdots\ c_1]\begin{pmatrix}a_1 & a_2 & \cdots & a_n\\a_2 & a_3 & \cdots & a_{n+1}\\\vdots & \vdots & & \vdots\\a_n & a_{n+1} & \cdots & a_{2n-1}\end{pmatrix}=[c_n\ c_{n-1}\ \cdots\ c_1]\boldsymbol{X}$$

若 $\boldsymbol{X}$ 可逆，则有 $[c_n\ c_{n-1}\cdots\ c_1]=[a_{n+1}\ a_{n+2}\cdots\ a_{2n}]\ \boldsymbol{X}^{-1}$。可以证明 $\boldsymbol{X}$ 的各列向量 $\boldsymbol{S}_1$，$\boldsymbol{S}_2$，…，$\boldsymbol{S}_{n+1}$ 线性无关，因此 $\boldsymbol{X}$ 可逆。

由此看到，采用线性移位寄存器产生的序列密码在已知明文攻击下是可以破译的。

【例 2-12】 设分析者得到的密文序列为 101101111010，相应的明文串为 100011011101，且他已知密钥流是由三级移位反馈寄存器产生的，试推导其反馈函数。

解： 由明密文序列，可计算出密钥流为 001110100111。由于所用移位寄存器是三级，所以解由开始的 6 个密钥位构成的矩阵方程：

$$[a_4\ a_5\ a_6]=[c_3\ c_2\ c_1]=\begin{pmatrix}a_1 & a_2 & a_3\\a_2 & a_3 & a_4\\a_3 & a_4 & a_5\end{pmatrix}$$

即

$$[1\ 1\ 0]=[c_3\ c_2\ c_1]=\begin{pmatrix}0 & 0 & 1\\0 & 1 & 1\\1 & 1 & 1\end{pmatrix}$$

所以

$$[c_3\ c_2\ c_1]=[1\ 1\ 0]\begin{pmatrix}0 & 0 & 1\\0 & 1 & 1\\1 & 1 & 1\end{pmatrix}^{-1}=[1\ 1\ 0]\begin{pmatrix}0 & 1 & 1\\1 & 1 & 0\\1 & 0 & 0\end{pmatrix}=[1\ 0\ 1]$$

因此产生密钥流的反馈函数为 $a_{i+3}=c_3a_i \oplus c_1a_{i+2}$ $(i \geqslant 1)$，结构如图 2-16 所示。

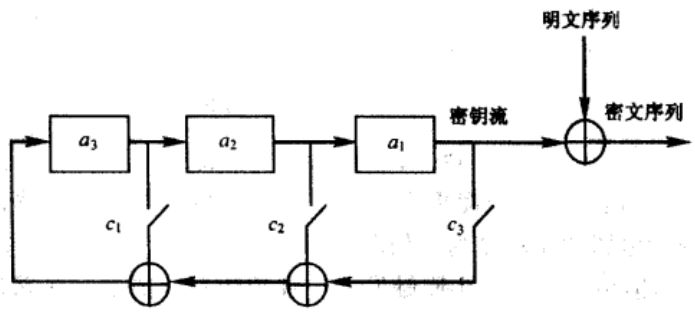


图 2-16 例 2-12 的密钥流反馈函数

（3）非线性序列

由于线性序列不适于用作密钥流，人们开始考虑非线性序列。目前非线性序列的产生主要有3种方式：一种是利用非线性反馈移位寄存器，第2种是利用LFSR输出驱动一个非线性函数，也称作非线性前馈序列生成器，第3种就是前面所讲的利用分组算法的密码反馈模式（CFB）和输出反馈模式（OFB）来产生序列密码。下面介绍前两种方法。

1）非线性反馈移位寄存器序列

当图2-13中的反馈函数$f(a_1, a_2, \cdots, a_n)$为非线性函数时，就构成了非线性反馈移位寄存器，因为n级反馈移位寄存器最多有2^{2^n}种，而线性寄存器占2^n种，因此n级非线性反馈移位寄存器有$2^{2^n}-2^n$种。非线性反馈移位寄存器的输出序列为非线性序列，序列的周期最大为2^n。周期达到最大值的非线性序列称为M序列。

定理2-30 在n级M序列的一个周期内：0与1的个数均为2^{n-1}；总游程数为2^{n-1}；对$1 \leqslant i \leqslant n-2$，长为$i$的游程数为$2^{n-1-i}$，其中0、1游程各一半；长为$n-1$的游程不存在；长为$n$的0游程和1游程各1个。

由此看出，M序列具有很好的随机统计特性。

定理2-31 GF（2）上n级M序列的数目为$2^{2^{n-1}-n}$个。

所以，M序列的数目还是很多的，它也是人们在流密码中的主要研究内容之一。

2）非线性前馈序列。

目前人们对一般的非线性反馈移位寄存器的研究还处于艰难的探索阶段，相比而言，利用LFSR的非线性组合来获得良好的非线性序列则较为简单。其基本思想是LFSR的输出不直接用作密钥流，而是用来驱动一个非线性组合函数所决定的电路，从而产生非线性序列。也就是说，将LFSR的输出前馈，因此也称其为非线性前馈序列生成器，所产生的非线性序列也叫做前馈序列，称非线性组合函数为前馈函数，如图2-17所示。

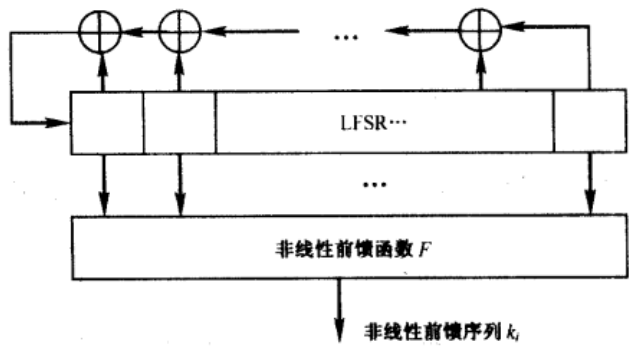


图2-17 非线性前馈序列生成器

以此为基础，人们提出了复杂度更高的非线性序列生成器设计方法，即用多个独立的LFSR的输出作为前馈电路的驱动序列，如图2-18所示。

典型的非线性前馈组合方式有J-K触发器、钟控序列生成器、多路选择发生器、多倍速率内积式发生器、自采样发生器等，从理论上来说，它们大部分都是良好的，但在实际应用中，绝大多数都是不安全的，许多看上去很复杂的非线性序列发生器均被破译了。

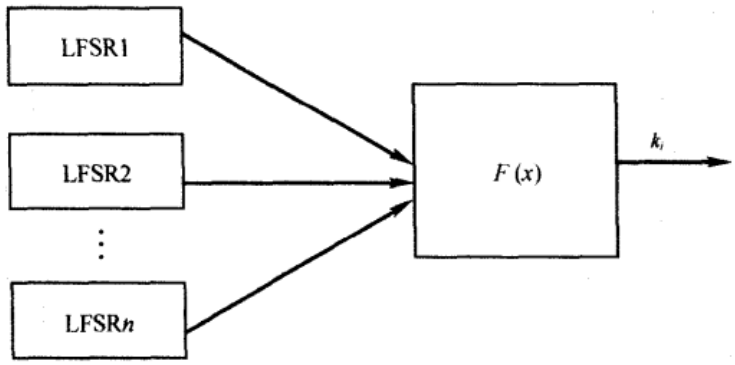


图 2-18　多 LFSR 的非线性前馈序列生成器

2.2.2　私钥密码与公钥密码

根据密钥的特点可将密码技术分为私钥密码技术和公钥密码技术。如一个密码体制的加密密钥与解密密钥相同或从其中一个能很容易导出另一个，则称为私钥密码体制，通常也称为单钥密码体制或对称密码体制；反之就是公钥密码体制，一般也叫非对称密码体制或双钥密码体制。

1. 私钥密码体制

在私钥密码体制中加密密钥和解密密钥相同（如图 1-4 所示），因此系统的保密性主要取决于密钥的安全性。私钥密码体制既可以采用分组加密技术，也可以采用流加密技术，如下面介绍的 DES 就是分组加密私钥密码体制。采用安全的私钥密码系统可以达到以下目的：

1）数据保密：明文经加密后只有拥有密钥的人才能正确地解密。

2）身份认证：这是通过传递随机数完成的。若通信的一方要认证另一方的身份，则他可以选择一个随机数，并要求对方加密后返回，若他解密后能得到原来的随机数，则可确定对方的身份无误。因为密钥只有他们双方知道。这种身份认证机制现在广泛应用于银行系统中。

3）数据完整性：发送方可将明文加密，并把密文附加于明文之后发送给接收方；接收方将所附的密文解密，对照解密后的信息是否与接收的明文一致，若一致说明明文没有被中途篡改，否则就有被改动的嫌疑。当然，接收方也可以将明文加密，对比加密后的密文与接收到的密文是否一致来确认数据的完整性。

以上这些都是建立在安全的密钥分发、存储基础上的。实际应用中，事先都要在收发双方之间建立一条安全的秘密信道来传递密钥。另外，在私钥密码体制中，密钥的数量太大。例如，若网络中有 n 个通信主体，那么每人必须拥有 $n-1$ 把密钥，系统中需要有 $n(n-1)/2$ 把不同的密钥，当 n 很大时，密钥的数量是惊人的。这时密钥的管理也是影响系统安全的重要因素。同时，私钥密码系统不具有防抵赖功能，因为密钥是双方共有的，所以任何一方伪造或篡改消息，另一方都拿不出证据来澄清事实。

正是由于私钥密码体制的上述缺点，人们开始寻求解决的办法，从而导致了公钥密码系统的诞生。

2. 公钥密码体制

公钥密码体制的思想是由 Diffie 和 Hellman 于 1976 年首先提出的，他们当时并没有实现一个公钥密码系统，但在他们的思想的影响下，许多著名的公钥密码系统都应运而生，并且

很多都得到了广泛的应用，如后面要介绍的 RSA 公钥系统。现今绝大多数的公钥密码体制都属于分组密码，概率密码体制除外。

在公钥密码系统中，各主体都拥有一对密钥，其中一个是秘密的，由他们本人产生、保管，称为私钥，用 K_S 表示；另一把则可以公开，称为公钥，表示为 K_P。一般公钥由相应的私钥计算得到，但从公钥却几乎推不出私钥。公钥密码体制最主要的特点是将加密与解密能力相分离，因此它具有下述优点：

1）可用于公共网络中的保密通信：由于某一主体的公钥可为许多人知道，因此他们都可以使用该公钥加密消息；但私钥只是私人拥有，因此只有知道对应私钥的主体可以解密这些密文信息。这样就实现了多个用户加密的消息只能由一个用户解读。

2）可达到不可否认功能：为使发送的消息具有认证性，可以用某主体的私钥加密消息，将此密文作为签名。由于私钥是保密的，因此任何人都不能伪造签名。同时任何接收方都可用对应的公钥解密签名信息，若得到有意义的明文，或该明文与接收到的明文一致，则可确定该消息的确是由拥有解密公钥的主体发送的。这个认证过程也称为数字签名。

假设用户 A 要向用户 B 发送消息，A 的公钥/私钥对为 K_{Pa}/K_{Sa}、B 的公钥/私钥对为 K_{Pb}/K_{Sb}，则上述两功能如图 2-19、图 2-20 所示。

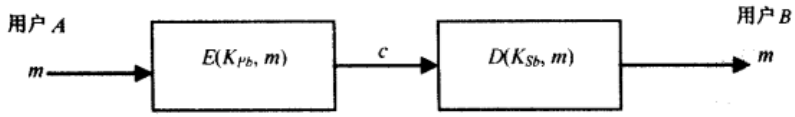


图 2-19　公钥保密功能

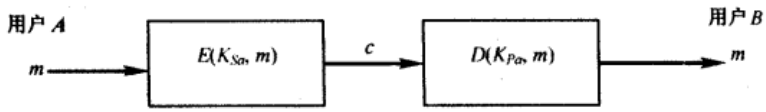


图 2-20　公钥认证功能

3）使用公钥密码体制，简化了密钥的分配与管理。因为在通信网络中的每个主体只需拥有一对私钥/公钥对即可，而且这些密钥都可由他们自己产生。与私钥密码体制相比，系统中的密钥量大大减少。

尽管公钥密码体制具有上述优点，但仍在某些方面颇遭非议，最主要的就是运算复杂，速度缓慢。为此，人们提出采用混合密码系统，即用公钥密码系统进行数字签名，达到密钥管理与身份认证功能，而用私钥密码系统对消息加密，确保通信机密性。另一方面，人们也在寻求更快的公钥密码技术，后面要介绍的椭圆曲线密码机制已引起了人们的普遍重视。

2.3　数据加密标准

数据加密标准（Data Encryption Standard，DES）的前身是 1971 年由 IBM 公司的 Horst Feistel 领导研制的 LUCIFER 算法，其密码设计思想 Feistel 网络充分体现了香农提出的混淆和扩散原则，DES 沿用了这一思想。LUCIFER 算法是分组密码算法，分组长度是 64bit，密钥长度为 128bit，一经推出就得到了很好的应用。之后，IBM 希望借此开发出一种商用加密产品，最好能在一块芯片上实现，Walter Tuchman 和 Carl Meyer 领导了该项目，对原 LUCI-

FER 算法进行改进，并将密钥长度缩短为 56bit，以便于芯片实现。

同时，在 20 世纪 60 年代末，随着计算机网络的发展，信息处理标准化要求日益强烈。为此，美国标准局（NBS，现国家标准与技术研究所 NIST）于 1973 年向社会公开征集标准密码算法，IBM 公司随后提交了改进的 LUCIFER 算法，经国家安全局（NSA）协助评估其安全性后，该算法于 1977 年被批准为数据加密标准（DES）。

DES 是迄今为止得到最广泛应用的一种密码算法，也是最有代表意义的分组加密体制。虽然它也受到了很猛烈的批评，而且随着 AES 的提出，它不会长期成为数据加密标准，但对它的基本原理、安全性分析、实际应用等进行较为深入的研究，对于掌握分组密码理论及相关领域的研究都是很有帮助的。

2.3.1 DES 算法描述

DES 的数据分组长度是 64bit，密文分组长度也是 64bit，不存在数据扩展问题。密钥长度是 64bit，但有 8bit 是奇偶校验位，因此有效密钥长度实际是 56bit。

DES 对 64bit 明文分组的处理可以概括为以下几步：首先进行初始置换（IP），将分组分为长度相等的左右两部分；然后进行 16 轮完全相同的运算，称为 f 函数，在每一轮中都要使用由初始密钥产生的子密钥；16 轮运算结束后，左右两部分合在一起进行末置换（初始置换的逆置换），得到加密结果，如图 2-21 所示。

（1）初始置换 IP

初始置换的目的是将 64bit 明文顺序打乱，得到一个乱序的 64bit 明文组，如表 2-1 所示。

表 2-1 IP 置换

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	53	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

表中各元素是明文中各比特的位置号数。可以看出其中的规律是各位置号数相差 8，即将原明文的各字节按列写出，各列比特经奇偶采样置换后，再对各行逆序。初始置换后的输出是将表中各元素按行读出。

（2）逆初始置换 IP^{-1}

逆初始置换将 16 轮迭代后的输出数据打乱重排，得到 64bit 密文输出，如表 2-2 所示。

表 2-2 IP^{-1} 置换

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

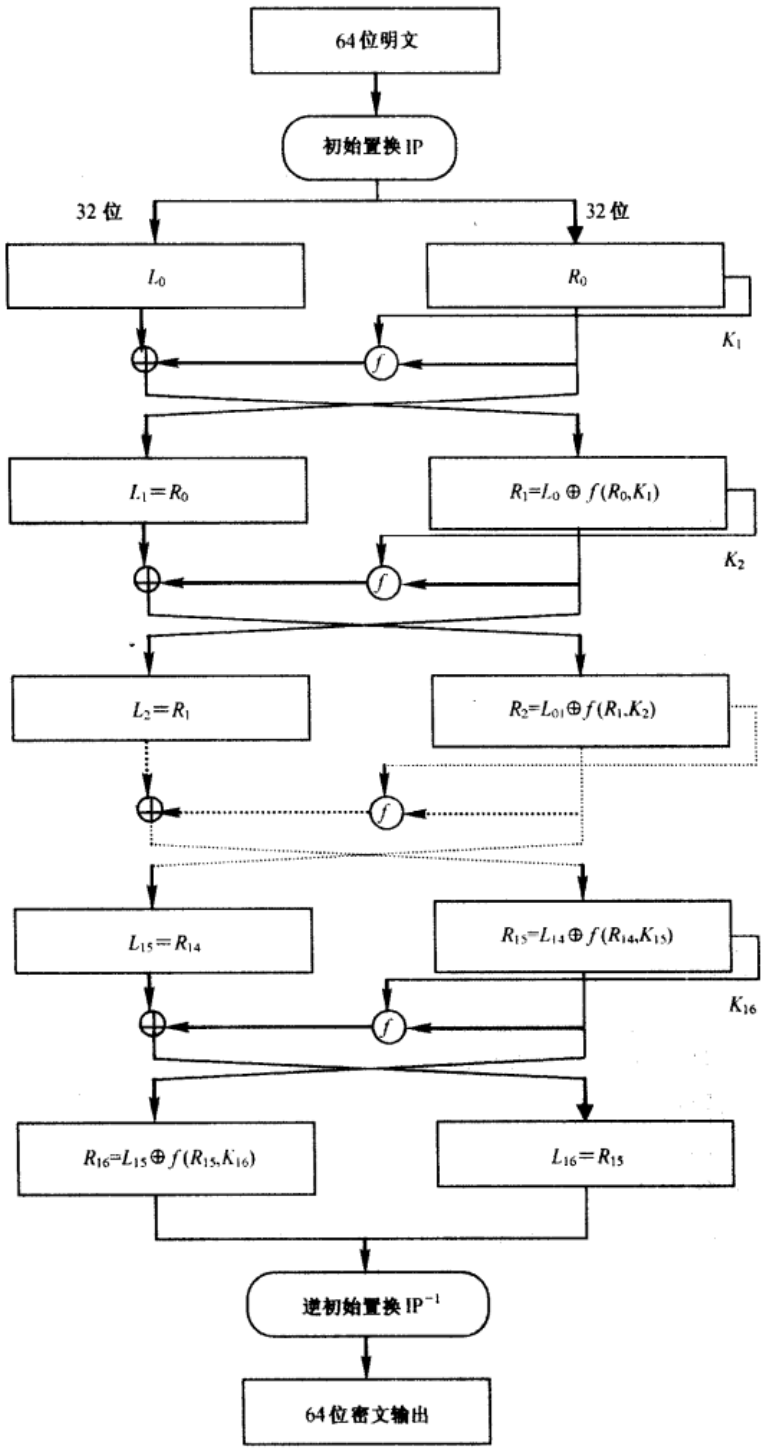

64位明文
初始置换IP
32位
32位
L_0
R_0
K_1
$L_1=R_0$
$R_1=L_0\oplus f(R_0,K_1)$
K_2
$L_2=R_1$
$R_2=L_{01}\oplus f(R_1,K_2)$
$L_{15}=R_{14}$
$R_{15}=L_{14}\oplus f(R_{14},K_{15})$
K_{16}
$R_{16}=L_{15}\oplus f(R_{15},K_{16})$
$L_{16}=R_{15}$
逆初始置换IP^{-1}
64位密文输出

图 2-21　DES 算法

之所以称其为逆初始置换，是因为若将一个64bit输入 M 经初始置换后，再进行逆初始置换，那么最终的输出仍是 M，即 $M = \mathrm{IP}^{-1}(\mathrm{IP}(M))$。

（3）产生子密钥

一个64bit明文组经初始置换后被分成左右两部分 L_0、R_0，各32bit，然后进行16轮迭代。首先由加密函数 f 对 R_0 加密，密钥为子密钥 K_1，得到 $f(R_0, K_1)$，$f(R_0, K_1)$ 再与 L_0 模2相加，得到 $L_0 \oplus f(R_0, K_1)$，以 $L_0 \oplus f(R_0, K_1)$ 作为第二次加密迭代的 R_1，以 R_0 作为第二次加密迭代的 L_1，第一次加密迭代过程结束。以后各次加密迭代的密钥分别为子密钥 K_2，K_3，…，K_{16}。可用数学公式描述为：

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \end{cases}, i = 1,2,\cdots,16 \tag{2-25}$$

最后一次的迭代 L_{16} 放在右边，R_{16} 放在左边。

每一个子密钥 K_i 是由原始密钥 K 经一系列置换选择后得到的，具体过程如图2-22。

因此，子密钥的产生主要有以下三个步骤：

1）置换选择1。

初始密钥是64bit，其中56bit是有效密钥，8bit是奇偶校验位，其位置号为8、16、24、32、40、48、56和64。置换选择1的作用是将8bit奇偶校验位去掉，把剩下的56bit打乱重排，并将前28bit放入 C_0，后28bit存入 D_0，如表2-3所示。

表2-3 密钥置换选择1

C_0	57	49	41	33	25	17	9
	1	58	50	42	34	26	18
	10	2	59	51	43	35	27
	19	11	3	60	52	44	36
D_0	63	55	47	39	31	23	15
	7	62	54	46	38	30	22
	14	6	61	53	45	37	29
	21	13	5	28	20	12	4

2）左循环移位置换。

在生成各子密钥时，C、D寄存器中的内容都要进行左循环移位置换，表中以 LS_i 表示，即

$$\begin{cases} C_i = LS_i(C_{i-1}) \\ D_i = LS_i(D_{i-1}) \end{cases}, \quad 1 \leqslant i \leqslant 16 \tag{2-26}$$

其中，LS_i 的值等于1或2，表示循环左移1位或2位，这取决于 i 的值。如果 $i = 1$，2，9，16就移一位，否则移两位。表2-4为密钥左循环移位次数表。

表2-4 密钥左循环移位次数表

第 i 次迭代	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
左移次数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

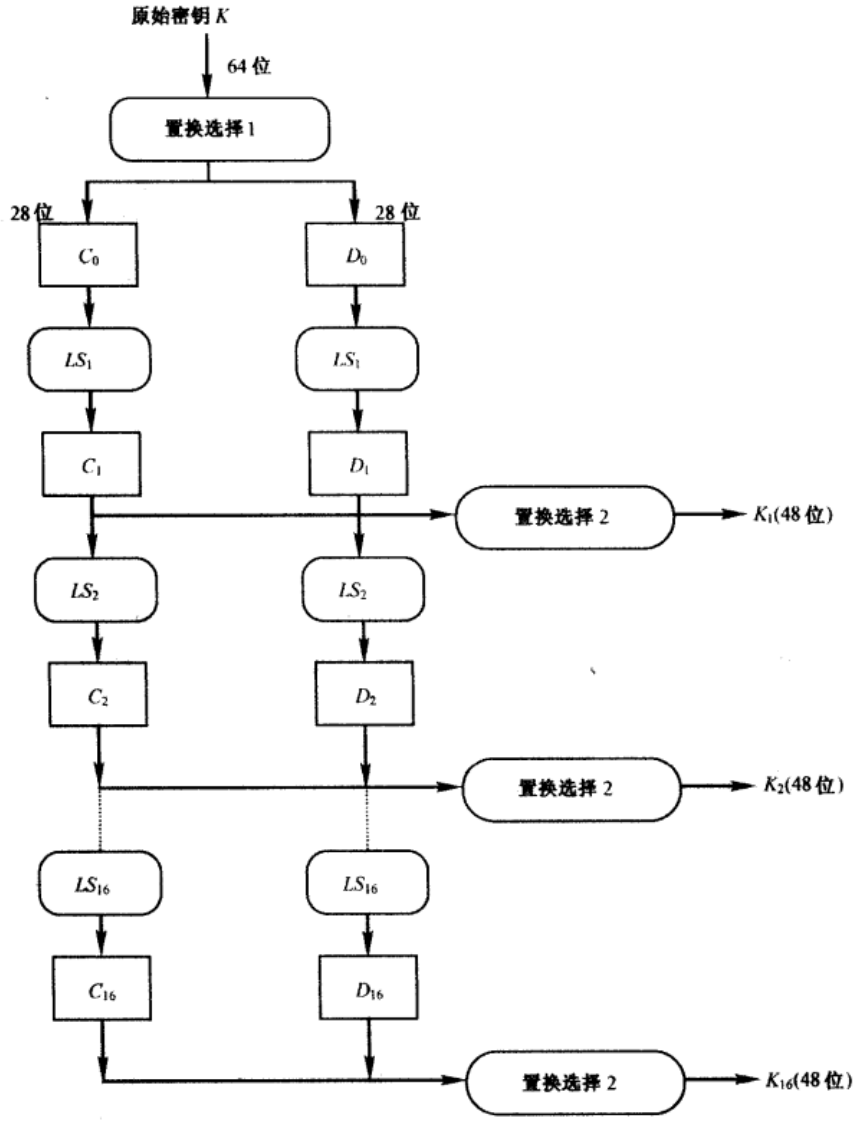


图 2-22　子密钥生成算法

3）置换选择 2。

经左循环移位后，C 和 D 寄存器中的内容还要进行一次置换选择，主要是将 C 中的第 9、18、22、25 位和 D 中的第 7、9、15、26 位删去，并将其余数字置换位置后输出，得到的 48bit 就是第 i 次迭代所用的子密钥 K_i，如表 2-5 所示。

表 2-5　密钥置换选择 2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

以上介绍了初始置换、子密钥生成以及逆初始置换，而 DES 的核心部分则是 f 加密函数，它的主要作用就是在第 i 次加密迭代中用子密钥 K_i 对 R_{i-1} 加密。

（4）f 函数

首先对 R_{i-1} 进行扩展置换 E，32bit 数据经选择、排列后，产生一个 48bit 的结果。该结果与子密钥 K_i 模 2 相加，然后送入 8 个 S 盒选择、代替，每个 S 盒有 6bit 输入，产生 4bit 输出，因此得到了一个 32bit 的数据组。该数据组再经过 P 盒置换，将其各位打乱、重排，得到的结果便为加密函数的输出 $f(R_{i-1}, K_i)$。图 2-23 描述了一轮 DES 的运算过程。

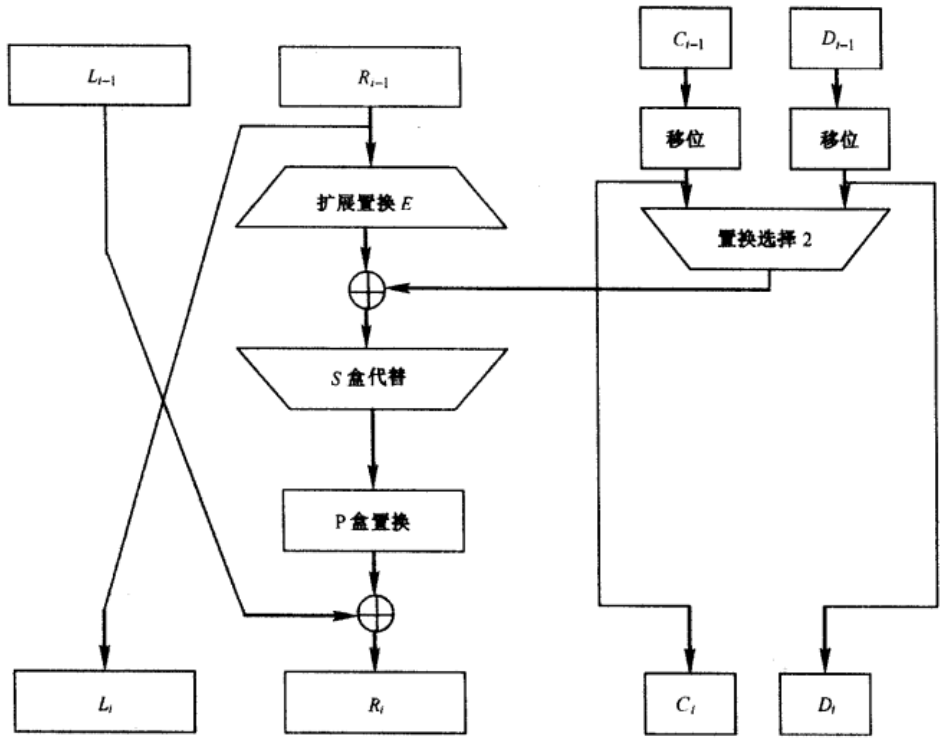


图 2-23　一轮 DES 运算

1）扩展置换 E。

该操作将输入的 32bit R_{i-1} 扩展成 48bit 输出，它改变了位的次序，重复了某些位，因此称为扩展置换。这样，一方面可以得到与密钥长度相同的数据，以进行异或运算；同时在下一次替代时可以对得到的较长结果进行压缩；最主要的是输入的一位可以影响两次替代，因此输出对输入的依赖性传播得更快，这就是雪崩效应，这样密文的每一位都可以更快地依赖于明文及密钥的每一位。扩展置换的选位如表 2-6 所示。

表 2-6　扩展置换

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

2）S 盒代替。

扩展置换后得到的48bit数据分组与子密钥异或，该结果被分成8个6bit分组，并行地送入8个 S 盒。每个 S 盒相当于一个置换选择函数，各有6个输入、4个输出。它们各用一个选择矩阵规定了输入与输出的选择规则，每个选择矩阵有4行16列，每一行都是0～15这16个数字，但每行数字的排列各不相同。8个选择矩阵也彼此不同，如表2-7所示。

表2-7 S 盒

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S_1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	12	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S_2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S_3
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S_4
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S_5
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S_6
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S_7
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S_8
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

选择规则为：取输入的二进制数的第1和第6位所组成的二进制数值所代表的行号，中间4位所组成的二进制数所代表的列号，处在所选中的行和列的交叉点的数字就是该 S 盒的输出。比如对于 $S3$，设输入为101011，第1位和第6位组成 $(11)_2 = (3)_{10}$，表示选中 $S3$ 的标号为3的那一行，其余4位 $(0101)_2 = (5)_{10}$，表示选中 $S3$ 标号为5的那一列，交叉点的数字为9，那么 $S3$ 的输出为1001。

3）P 盒置换。

对 S 盒输出的32bit数据直接进行置换，任何一位既不能被略去也不能被映射多次。P

盒置换如表 2-8 所示。

表 2-8　*P* 盒置换

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

P 盒的输出结果与上一轮的左半部分异或，然后左右两部分对换，开始下一轮迭代。

DES 在经过所有的代替、置换、循环移位及异或运算后，解密操作仍可以使用相同的算法，惟一的不同之处是密钥的使用次序相反。也就是说，解密时把 64bit 密文作为输入，第一次解密迭代使用密钥 K_{16}，第二次解密迭代用 K_{15}，依此类推，第 16 次解密迭代用 K_1，最后输出的就是明文。可用数学公式描述为：

$$\begin{cases} R_{i-1} = L_i \\ L_{i-1} = R_i \oplus f(L_i, K_i) \end{cases}, \quad i = 16, 15, \cdots, 1 \tag{2-27}$$

2.3.2　DES 算法的实现

虽然关于 DES 的描述很长，但它能以硬件或软件的方式非常有效地实现，主要的算术运算是比特串的异或，各种置换操作及子密钥的生成都可以通过硬布线（硬件电路）或查表操作（软件实现）在固定的时间内完成。自 DES 正式颁布以后，世界上许多公司都设计并推出了它们自己实现的 DES 的有关软硬件产品。表 2-9 列出了一些商用 DES 芯片，表 2-10 列出了采用软件实现，在不同微处理器上的运算速度。

目前除了一些敏感的政府部门以外，DES 在许多政府部门及非政府部门都得到了广泛的应用。其中一个重要的应用领域就是银行交易，如采用 DES 算法加密个人身份识别码（PIN）及通过自动取款机（ATM）的交易。

表 2-9　商用 DES 芯片

（摘自 Applied Cryptography—Protocols, algorithms, and source code in C）

制造商	芯片	制造日期	时钟 /MHz	数据速率 /（Mbit/s）	可用性
AMD	Am9518	1981	3	1.3	否
AMD	Am9568	?	4	1.5	否
AMD	AmZ8068	1982	4	1.7	否
AT&T	T7000A	1985	?	1.9	否
CE-Infosys	SuperCrypt CE99C003	1992	20	12.5	是
CE-Infosys	SuperCrypt CE99C003A	1994	30	20.0	是
Cryptech	Cry12C102	1989	20	2.8	是
Newbridge	CA20C03A	1991	25	3.85	是
Newbridge	CA20C03W	1992	8	0.64	是
Newbridge	CA95C68/18/09	1993	33	14.67	是
Pijnenburg	PCC100	?	?	2.5	是

（续）

制造商	芯片	制造日期	时钟 /MHz	数据速率 /（Mbit/s）	可用性
Semaphore Communications	Roadrunner284	?	40	35.5	是
VLSI	VM007	1993	32	200.0	是
VLSI	VM009	1993	33	14.0	是
VLSI	6868	1995	32	64.0	是
Western Digital	WD2001/2002	1984	3	0.23	否

表 2-10　DES 软件实现速度

（摘自 Applied Cryptography—Protocols，algorithms，and source code in C）

处理器	速度/MHz	DES 分组/s
8088	4.7	370
68000	7.6	900
80286	6	1100
68020	16	3500
68030	16	3900
80286	25	5000
68030	50	10000
68040	25	16000
68040	40	23000
80486	66	43000
Sun ELC		26000
HyperSparc		32000
RS6000-350		53000
Sparc 10/52		84000
DEC Alpha 4000/610		154000
HP 9000/887	125	196000

2.3.3　DES 的安全性分析

自从 DES 诞生以来，人们对其安全性的置疑就从没有停止过。大量的研究主要集中在两个方面：一是 DES 算法本身的特点和性质，二是对 DES 密码体制的攻击。

1. DES 算法的性质

（1）DES 具有很强的雪崩效应

对于加密算法，希望明文或密钥某一位的变化能够引起密文的很多位发生变化，否则就可能缩小有待搜索的明文和密钥空间的大小为密码分析者提供可乘之机。研究表明，DES 具有很强的雪崩效应。Konheim 给出了用同一密钥加密两组仅差一位的明文及用两个相差一位的密钥加密同一明文分组时，不同的密文比特在各轮中的个数。表 2-11 列出的是用密钥：

0000001 1001011 0100100 1100010 0011100 0011000 0011100 0110010

加密两组只相差一位的明文：

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

表 2-11　明文变化时 DES 的雪崩效应

迭代轮数	0	1	2	3	4	5	6	7	8
不同的密文比特数	1	6	21	35	39	34	32	31	29
迭代轮数	9	10	11	12	13	14	15	16	
不同的密文比特数	42	44	32	30	30	26	29	34	

表 2-12 列出的结果是采用两个仅相差一位的密钥：

1110010 1111011 1101111 0011000 0011101 0000100 0110001 0010111

0110010 1111011 1101111 0011000 0011101 0000100 0110001 0010111

加密一明文：

01101000 10000101 11011100 01111010 00010011 01110110 11101011 10100100

表 2-12　密钥变化时 DES 的雪崩效应

迭代轮数	0	1	2	3	4	5	6	7	8
不同的密文比特数	0	2	14	28	32	30	32	35	34
迭代轮数	9	10	11	12	13	14	15	16	
不同的密文比特数	40	38	31	33	28	26	34	35	

可以看出在短短三轮之后就表现出了雪崩效应，而且大约一半的密文比特都不同。

（2）DES 存在弱密钥及半弱密钥

在 DES 加密中每一轮都采用一个不同的子密钥，各子密钥是由初始密钥产生的，这也是用来加强 DES 安全强度的一个重要措施。但也有些密钥，由它产生的 16 个密钥是完全相同的，这样的密钥称为弱密钥。同样，有些密钥只产生两种不同的子密钥，每种子密钥各用 8 次，这样的密钥称为半弱密钥。类似的还有四分之一弱密钥等。

这些弱密钥存在的原因很简单。在生成子密钥时，把初始密钥分成了两部分，每部分是独立地移位。如果两部分的所有位都是 0 或 1，那么算法在任一周期的子密钥都是相同的；若一部分全为 1，另一部分全为 0，就只能产生两种子密钥。

对于弱密钥 k，它所产生的各子密钥为 $k_1 = k_2 = \cdots = k_{16} = k$。对于弱密钥有：

$$\mathrm{DES}_k(\mathrm{DES}_k(x)) = x \tag{2-28}$$

$$\mathrm{DES}_k^{-1}(\mathrm{DES}_k^{-1}(x)) = x \tag{2-29}$$

也就是用 k 对明文 x 加密两次或解密两次都可以恢复出明文，加密操作和解密操作没有区别。因此，弱密钥使得 DES 在选择明文攻击下的搜索量减半。

半弱密钥是成对出现的，若 k_1 和 k_2 是一对互逆的半弱密钥，则有：

$$\mathrm{DES}_{k_2}(\mathrm{DES}_{k_1}(x)) = \mathrm{DES}_{k_1}(\mathrm{DES}_{k_2}(x)) \tag{2-30}$$

半弱密钥也使 DES 的安全性变差。

下面分别以 16 进制列出所存在的 DES 的弱密钥及半弱密钥（包括 8bit 奇偶校验位），如表 2-13 和表 2-14 所示。

表 2-13　DES 弱密钥

0101	0101	0101	0101
1F1F	1F1F	0E0E	0E0E
E0E0	E0E0	F1F1	F1F1
FEFE	FEFE	FEFE	FEFE

表 2-14　DES 半弱密钥

01FE	01FE	01FE	01FE
FE01	FE01	FE01	FE01
1FE0	1FE0	0EF1	0EF1
E01F	E01F	F10E	F10E
01E0	01E0	01F1	01F1
E001	E001	F101	F101
1FFE	1FFE	0EFE	0EFE
FE1F	FE1F	FE0E	FE0E
011F	011F	010E	010E
1F01	1F01	0E01	0E01
E0FE	E0FE	F1FE	F1FE
FEE0	FEE0	FEF1	FEF1

在表 2-14 中，表格中每一行的上下两个密钥分别构成了互逆对。

虽然弱密钥及半弱密钥对 DES 的安全性构成了威胁，但它们的数量与总的密钥量相比是微不足道的，只要在实际应用中注意不要使用这样的密钥即可。

（3）DES 具有互补对称性

将密钥 K 的每一位取反，也就是用 0 代替 1，用 1 代替 0，就得到了该密钥的补密钥 $\overline{K}$，同样可以得到某一明文 M 的补 $\overline{M}$ 或某一密文 C 的补 $\overline{C}$。对于 DES，若 $DES_K(M)=C$，则 $DES_{\overline{K}}(\overline{M})=\overline{C}$。也就是说，若用原密钥加密一明文分组得到一个密文分组，那么用该密钥的补密钥加密该明文分组的补得到的是原密文分组的补。互补对称性使得对 DES 的选择明文攻击工作量减半。

DES 具有互补对称性是因为它在每一轮计算中用到了两次异或运算，同时对于异或运算有以下特性：若 $y=x_1\oplus x_2$，则 $y=\overline{x}_1\oplus\overline{x}_2$。

2. 对 DES 的安全攻击方法

（1）穷举攻击

对 DES 的较为一致的看法就是它的密钥长度短了些。DES 的密钥空间大小为 $2^{56}=7.2\times10^{16}$。若对此进行密钥搜索破译，分析者在得到一组明文/密文对的条件下，可对明文用不同的密钥加密，直到得到的密文与已知的密文相同，这样就确定出了所用的密钥，为此人们提出了密钥搜索机的想法。

在 1977 年，Diffie 等人证明一台专用于破译 DES 的并行计算机能在一天中找到密钥，但需耗资 2000 万美元；1993 年，Michael Wiener 设计了一个 100 万美元的机器，它能在平均 3.5 小时内完成对 DES 的穷举攻击。不过以上这些只是一些假想，并没有实现真正的机器。RSA 公司在 1997 年提供了 1 万美元的奖金，通过 Internet 搜索 DES 密钥，项目实施 96 天后

找到了正确的密钥，有7万多个系统参与了这项工作，搜索了大约1/4的可能密钥，这也显示了分布式个人计算机在密码分析时的威力。到1998年，美国EFF（Electronic Frontier Foundation）宣布他们将一台20万美元的计算机改装成了专用解密机，用56小时破译了DES。

（2）差分密码分析

差分密码分析在1990年才作为一种新的密码分析方法公之于众，它是由Eli Biham和Adi Shamir首次提出的。利用这种方法，他们找到了一个比穷举攻击更有效的选择明文攻击DES的方法，他们称在有2^{47}个选择明文的情况下，可以以小于2^{55}的复杂性成功破译DES。同时该分析方法也影响了后续一些分组密码算法的设计，如IDEA等。因此，差分密码分析方法的提出是密码分析方面最重要的进展之一。

差分密码分析考察的是那些明文有特定差分的密文对，分析当采用相同的密钥加密时，该差分在各轮中的变化（注：差分运算可以有不同的定义，在DES中定义为异或运算）。首先任意选取具有固定差分的一对明文，然后使用输出密文的差分，按照不同的概率分配不同的密钥，随着分析的密文对不断增多，就可以分析出正确的密钥。具体的分析过程如图2-24所示。

其基本思路为假设知道一对输入X和X'及相应的密文输出Y和Y'，那么它们的差分ΔX和ΔY也是已知的。由于扩展置换是已知的，那么ΔA也是已知的。虽然我们还不知道K_i，但差分ΔB等于ΔA。同时S盒也是已知的，因此ΔC也是已知的。将ΔA和ΔC联合起来，就可以推出K_i的信息。通过差分分析，得到的是48bit密钥的值，剩下的8bit可以通过穷举攻击得到。

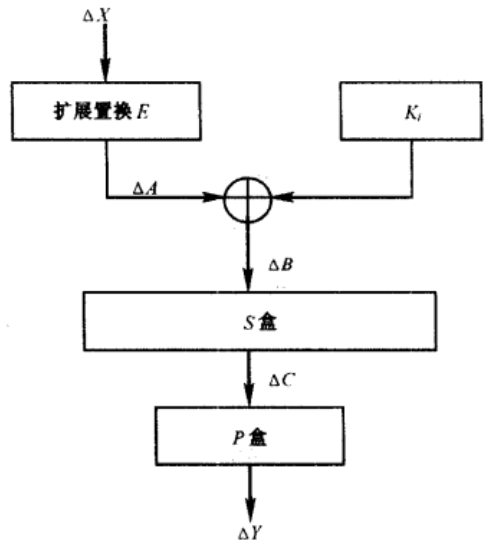


图2-24　DES差分分析

差分分析极大地依赖于S盒的设计，而分析表明，DES的S盒能够很好地抵抗差分攻击。同时，由于差分分析所需要的巨大的时间量和数据量，它只具有理论上的意义。

（3）线性密码分析

线性密码分析是由Mitsuru Matsui提出的另一种密码分析攻击方法，它利用线性近似值来描述DES的操作。采用线性密码分析可以在有2^{47}个已知明文的情况下破译一个DES密钥。

假设有一对$n$bit的明文/密文对$M[1], M[2], \cdots, M[n]$ /$C[1], C[2], \cdots, C[n]$和一个$m$bit的密钥$K[1], K[2], \cdots, K[m]$，定义

$$A[i,j,\cdots,k] = A[i] \oplus A[j] \oplus \cdots \oplus A[k]$$

线性密码分析就是找到一个有效的线性方程：

$$M[\alpha_1,\alpha_2,\cdots,\alpha_A] \oplus C[\beta_1,\beta_2,\cdots,\beta_B] = K[\gamma_1,\gamma_2,\cdots,\gamma_D]$$

使其以概率$p \neq 0.5$成立。其中，α、β、γ表示某一比特位置。

线性分析也极大地依赖于S盒的设计，而DES所选择的S盒只具有很小的抗线性分析的能力。在12台HP9000/735工作站上，采用软件实现，花了50天的时间就成功攻击了完

整的 16 轮 DES。

（4）相关密钥密码分析

相关密钥密码分析是第一个攻击 DES 子密钥产生算法的分析方法，它考察的是不同密钥间的差分。在进行选择明文攻击时，密码分析者选择明文，并用两个不同的密钥加密。该方法与密码算法的迭代轮数无关，但由于 DES 密钥的循环移位是变化的，因此该攻击方法并不实用。

除了以上密码分析方法外，人们也进行了高阶差分密码分析、差分—线性分析等，在此不再详述。

2.4 RSA 密码算法

RSA 算法是用它的 3 个发明者 Rivest、Shamir 和 Adleman 的名字的首字母命名的。它是基于数论中欧拉定理而设计的一个公钥密码系统，其安全性取决于大整数因子分解的困难性，而后者被公认为是一个数学难题，虽然其困难性并没有得到严格的证明。从另一个角度来说，RSA 算法是一个分组密码算法。相对于某个数 n 来说，分组的大小必须不大于 $\log_2 n$，即明文字母表和密文字母表均取 $Z_n = \{0, 1, \cdots, n-1\}$。

该算法自公布以后，就逐步得到了广泛应用，是一个最流行的公钥加密算法，同时它还经受住了人们多年的深入分析，是一个比较可靠的密码算法。

2.4.1 RSA 加密算法描述

现在先来看一下在一个 RSA 加密体制中，某一个用户 i 的公钥及私钥的生成。

首先他随机地选取两个大素数 p_i 和 q_i，并计算

$$n_i = p_i q_i \tag{2-31}$$

及其欧拉函数值

$$\varphi(n_i) = (p_i - 1)(q_i - 1) \tag{2-32}$$

然后他随机地选取一个整数 e_i，满足 $1 \leqslant e_i < \varphi(n_i)$，且 $(e_i, \varphi(n_i)) = 1$。因此在模 $\varphi(n_i)$ 下，e_i 有逆元。可以利用欧几里得算法计算 d_i，使得

$$d_i e_i = 1 \bmod \varphi(n_i) \tag{2-33}$$

至此，用户 i 就可以公布 (n_i, e_i)，将其作为公钥；而 d_i 是私钥，予以保密，p_i 和 q_i 也要保密，或者立刻销毁。

相应地，加密算法为

$$c = E(e_i, m) = m^{e_i} \bmod n_i \tag{2-34}$$

而解密算法为

$$m = D(d_i, c) = c^{d_i} \bmod n_i \tag{2-35}$$

只要能够证明由解密运算可以恢复出明文，就可以证明该加/解密机制是正确的。证明过程很简单：

$$\begin{aligned} D(d_i, c) &= c^{d_i} \bmod n_i \xlongequal{(2\text{-}34)} (m^{e_i})^{d_i} \bmod n_i \xlongequal{(2\text{-}33)} m^{k\varphi(n_i)+1} \bmod n_i \\ &= (m^{k\varphi(n_i)} m) \bmod n_i \end{aligned}$$

若（m，n_i）$=1$，则由欧拉定理 $m^{\varphi(n_i)} = 1 \bmod n_i$，可得上式 $= m \bmod n_i = m$。

若（m，n_i）$\neq 1$，因为 $n_i = p_i q_i$，所以（m，n_i）必含 p_i 或 q_i，不妨设为 p_i，即（m，n_i）$= p_i$，则有 $m = cp_i$，$1 \leqslant c < q_i$，故

$$m^{\varphi(q_i)} = 1 \bmod q_i,\ m^{k\varphi(q_i)(p_i-1)} = 1 \bmod q_i$$

因此

$$m^{k\varphi(n_i)} = 1 + aq_i \text{ 和}$$

$$m^{k\varphi(n_i)+1} = m(1 + aq_i) = m + acq_i p_i = m + acn_i$$

所以

$$m^{k\varphi(n_i)+1} = m \bmod n_i = m$$

由上看到，RSA 算法的陷门在于模指数函数的单向性。即已知 m 和 e，求解 $c = m^e \bmod n$ 很容易，但若已知 c 和 e，求解 m，使之满足 $c = m^e \bmod n$（或 $m = \sqrt[e]{c} \bmod n$）则几乎是不可能的。这等同于对整数 n 的因子分解问题，因为一旦我们知道了 n 的素因子分解，计算上述逆问题就是可行的了。

采用 RSA 加密算法的加/解密过程可描述如下：

假设系统中用户 B 要将明文 m 加密后发给用户 A。那么用户 B 先从公钥数据库中查得用户 A 的公钥（n_A，e_A），然后将 m 分组为 $m = m_1 m_2 \cdots m_r$，其中每一个分组 $m_i \in Z_{n_A}$，最后对每一个分组分别加密 $c_i = m_i^{e_A} \bmod n_A$，并将密文 $c = c_1 c_2 \cdots c_r$ 发给用户 A。

而当用户 A 接收到密文 $c = c_1 c_2 \cdots c_r$ 后，将对每一组密文分别做解密运算 $m_i = c_i^{d_A} \bmod n_A$，最后合并各解密分组就得到了 B 发给他的明文。

【例 2-13】 用户 A 与 B 要采用 RSA 密码体制进行保密通信，B 发送消息 $m = 5$ 至 A，则加密与解密过程如下：

1）若用户 A 选取 $p = 13$，$q = 17$，随机选取密钥 $e = 71$，则 $n = 13 \times 17 = 221$，$\varphi(n) = 12 \times 16 = 192$，利用扩展欧几里得算法求解密钥 $d = e^{-1} \bmod \varphi(n) = 71^{-1} \bmod 192 = 119$。A 将 e 和 n 公开，保密 p、q 和 d。

2）用户 B 将消息明文分组 m，采用 A 的公钥 e 加密成密文分组 c：

$c = m^e \bmod n = 5^{71} \bmod 221 = 112$，并将 c 发送给 A。

3）用户 A 在接收到密文 c 后，使用私钥 d 进行解密变换：

$$m' = c^d \bmod n = 112^{119} \bmod 221 = 5 = m$$

2.4.2 RSA 数字签名

在前文提到采用公钥密码体制的一个重要优点就是可以通过数字签名机制达到身份认证、防抵赖等目的。

假设用户 A 要将一个消息 m 及其签名一起发给 B，A 用他的私钥对 m 签名为

$$S = m^{d_A} \bmod n_A \tag{2-36}$$

B 接收到（m，S）后，首先验证 S 是否正确，即验证

$$S^{e_A} \bmod n_A = m \tag{2-37}$$

是否成立，若成立，则 S 是 m 的签字，否则认为该消息不可信。

由于签名是采用签名者的私钥进行的，而该私钥是保密的，因此任何人都不能伪造签名；另一方面，对签名的验证采用签名者的公钥，因此易于证实该签名的合法性。

2.4.3 RSA 算法的实现

目前有许多用于 RSA 加密的芯片，但硬件实现的 RSA 的速度只有 DES 的 1/1000，虽然在智能卡中已大量实现了 RSA 算法，但速度都较慢。

软件实现的 RSA 的速度也只有 DES 的约 1/100。无论技术如何发展变化，RSA 的速度永远也比不上单钥密码算法的速度。

因此人们一般采用 RSA 算法加密一些信息量较小的关键信息，如密钥等，而在一些要求快速通信或传递信息量很大的场合仍然使用私钥加密算法。

2.4.4 RSA 的安全性分析

人们对 RSA 密码体制的安全性研究主要集中在以下几个方面。

1. 大整数的因子分解

RSA 的安全性依赖于大数分解的难度，分解大整数 n 也是一个最直接、最显然的攻击方法。有些密码分析者还会考虑到猜测每一个可能的私钥值，进行穷举攻击，实际上这种方法还没有对 n 进行因子分解快。迄今为止，人们提出的因子分解算法有试除法、Pollard 算法、椭圆曲线法、二次筛选法、数域筛选法等。随着计算能力的不断增长以及因子分解技术的不断改进，RSA 所受到的安全威胁也越来越大，人们不得不采用更大的密钥。在第 4 章将再一次涉及到这个问题。

2. 针对协议的攻击

有些密码分析者不是攻击基本的密码算法，而是通过攻击采用 RSA 算法的密码协议来达到破译消息的目的。

例如，一个攻击者窃听用户 A 的通信链路，他获得了以用户 A 的公钥 e 加密的密文 c，并且知道该用户的公钥 (n, e)，想分析出明文 m。首先他随机选取一个小于 n 的整数 r，计算 $y_1 = r^e \bmod n$，因此有 $r = y_1^d \bmod n$；接着计算 $y_2 = y_1 y \bmod n$，$t = r^{-1} \bmod n$，则有 $t = y_1^{-d} \bmod n$。一切准备好后，该窃听者就请 A 对 y_2 签名，得到 $S = y_2^d \bmod n$。计算 $tS \bmod n$，就可得到明文消息 m。这是因为 $tS \bmod n = y_1^{-d} y_2^d \bmod n = y_1^{-d} y_1^d y^d \bmod n = y^d \bmod n = m$。

上述攻击之所以能够凑效是因为指数运算保持了输入的乘法结构。为防止此类攻击，在对消息签名之前，都要先用一个单向散列函数对消息进行散列运算。

3. 针对 RSA 实现的攻击

这类攻击仍然是避开了对基本算法的攻击，而是针对 RSA 算法的实现细节进行攻击。

(1) 公共模数攻击

若在实现 RSA 算法时，很多人共用同一个 n，而选择不同的 e 和 d，则会产生系统的安全隐患。这是因为若采用两个互质的密钥加密同一个消息，则不需要任何解密密钥就可以恢复出该消息。一种可行的方法如下：

设共同的模数为 n，两加密密钥 e_1 和 e_2 互质，它们将加密同一个消息 m，则两个密文分别为 $c_1 = m^{e_1} \bmod n$，$c_2 = m^{e_2} \bmod n$，同时攻击者已经知道了 n、e_1、e_2、c_1 和 c_2。因为 e_1 和 e_2 互质，由欧几里得算法可找到 r 和 s，满足 $re_1 + se_2 = 1$。r 和 s 中必有一个为负数，不妨假设 $r < 0$。然后再用欧几里得算法计算出 c_1^{-1}，至此可得：

$$(c_1^{-1})^{-r}c_2^{s} = m^{re_1+se_2} = m$$

为防止公共模数攻击，不要在用户间共享 n。

（2）低加密指数攻击

若采用小的 e 可以加快加密和验证签字的速度。但研究表明，若 e 选得太小，则会影响 RSA 系统的安全性。

假设系统中有 3 个用户的 e 均选 3，但有不同的模 n_1、n_2 和 n_3，那么另一用户将同一消息 m（$m<n_1$，$m<n_2$，$m<n_3$）传送给这 3 个用户的密文分别为

$$c_1 = m^3 \bmod n_1, c_2 = m^3 \bmod n_2, c_3 = m^3 \bmod n_3$$

一般来说 n_1、n_2、n_3 互质，利用中国剩余定理可求出 $c = m^3 \bmod (n_1 n_2 n_3)$，又因为 $m<n_1$，$m<n_2$，$m<n_3$，所以 $m^3 < n_1 n_2 n_3$，故 $\sqrt[3]{c} = m$。

防止该攻击的一个简单方法就是用独立随机值来填充消息。

另外，若 d 取得过小，也会造成安全攻击，一般要选择大一点的 d。

（3）迭代攻击

迭代攻击是基于以下定理来破译 RSA 算法的。

定理 2-32 设 n 为素数或不同素数之积，若 $(e, \varphi(n)) = 1$，则 $a^{e^h} \equiv a \bmod n$，其中，h 为正整数。

假设密码分析者获得了密文 c 及加密公钥 (n, e)，他可以对密文利用公钥依次进行如下变换：

$$c^e \equiv c_1 \bmod n, c_1^e \equiv c_2 \bmod n, \cdots, c_{\beta-1}^e \equiv c_\beta \bmod n, c_\beta^e \equiv c_{\beta+1} = c \bmod n$$

也就是经过 $(\beta+1)$ 步迭代后，运算结果就是密文，那么在第 β 步的变换结果恰好就是明文 m。

比如，若 $(n, e) = (35, 17)$，密文为 $c=3$，那么 $c_1 = 3^{17} \bmod 35 = 33$，$c_2 = 33^{17} \bmod 35 = 3 = c$，所以明文为 $m = c_1 = 33$。

为防止迭代攻击，需要 β 足够大，Rivest 等人证明当 $(p-1)$ 和 $(q-1)$ 不具有小的素因子，且 n 足够大时，就可以保证 β 也足够大，从而使得 RSA 在实际上仍是不可破译的。

4. 定时攻击

这种攻击方式依赖于解密算法的运行时间，它通过测定 RSA 解密所用的模指数运算时间来估计私钥 d。采用这种攻击方式只需用到密文，类似于一个小偷通过观察某人转动拨号盘得到一个个数字的时间来猜测一个密码箱的密码数字组合。

为防止该攻击，可以采取一些简单的措施，如常数取幂时间、随机延时、盲化等。其中，常数取幂时间是保证所有的取幂操作在返回一个结果之前花费同样多的时间，这使得算法的性能下降；随机延时方法是对取幂算法增加一个随机延时来迷惑定时攻击者，不过要增加足够多的噪声，否则攻击者仍然可以通过收集更多的测量结果来成功地补偿随机延时；盲化操作从根本上防止了定时攻击的逐位分析，主要是在取幂运算之前先用一个随机数与密文相乘。

2.5 小结

作为全书的基础，本章从多个方面介绍了现代密码技术的相关基础知识及基本原理，作

为现代密码技术的典型代表，我们分别介绍了对称密码算法 DES 和公钥密码算法 RSA。

2.6 习题

1. 试证明公式 2-12、2-13、2-14。
2. 简述分组密码的几种不同运行模式，试分析其优缺点。
3. 熟悉 DES 加密方案并编程实现对单一分组的加密模块。
4. 在习题 3 实现的功能模块的基础上，编程实现 DES 在 4 种操作模式下的加/解密方案。
5. 求 gcd（216 793，256 027）。
6. 求 21017 模 25139 的乘法逆元。
7. 试验证 2 是模 25 的一个本原元。
8. 试判断 69 是否是模 101 的平方。
9. 求一个整数 x，使得 $x = 1 \bmod 12$ 且 $x = 8 \bmod 35$。

第3章　先进加密标准

我们在第2章看到，DES的56位密钥太少，使其易受穷举攻击，为此人们致力于寻找一种强度更高的对称密码算法。采用的方法主要有两种，一种是用DES和多个密钥进行多次加密，如得到广泛使用的三重DES；另一种是设计一个新的算法。采用第一种方法在提高算法安全性能的同时，可以保护已有的软件和设备投资，但加/解密的速度并不是很理想。人们也设计了很多对称分组密码算法，如IDEA、Blowfish、RC5等，它们都远不如DES普及。

美国国家标准技术研究所（NIST）于1997年4月15日发起了征集先进加密标准AES（Advanced Encryption Standard）的活动，并于1997年9月12日在联邦登记处（FR）公布了征集AES候选算法的通告，目的是确定一个非保密的、公开批露的、全球免费使用的分组密码算法，用于保护21世纪政府的敏感信息，并希望能够成为秘密和公开部门的数据加密标准。1998年8月20日，NIST召开了第一次AES候选会议，并公布了15个符合基本要求的候选算法，1999年3月22日，NIST举行了第二次AES候选会议，从15个候选算法中筛选出5个候选者，2000年4月25日，NIST举行了第三次AES候选会议，对这5个候选算法又进行了讨论。本章将较为详细地分析这5个AES候选算法。2000年10月2日，NIST公开了最终评选结果，将Rijndael算法作为AES标准算法。2001年11月26日，NIST正式公布高级加密标准AES，并于2002年5月26日正式生效。

3.1　AES简介

对AES的基本要求是比三重DES快而且至少和三重DES一样安全，分组长度为128位，密钥长度为128/192/256位可选。NIST对AES进行评估的主要准则是安全性、效率和算法的实现。其中安全性是第一位的，算法应能抵抗已有的密码攻击方法和可能的密码分析方法。在保证安全性的条件下，效率是最重要的评估因素，包括算法在不同平台上的计算速度和对内存的需求等。算法的实现主要是指其灵活性，如算法可以用软件和硬件实现，可以作为序列密码、杂凑算法实现，在不同的环境中都能够有效地实现和运行等。

NIST最初公布的15个候选算法的基本情况如表3-1所示。

表3-1　15个AES候选算法简表

序号	算法名称	原有算法	申　请　人
1	CAST-256	CAST-128	Entrust Technologies，Inc
2	CRYPTON	—	Future Systems，Inc
3	DEAL	DEA	Richard Outerbridge，Lars Knudsen
4	DFC	—	Centre National pour la Recherche Scientifique（CNRS）
5	E2	—	Nippon Telegraph and Telephone Corporation（NTT）
6	FROG	—	TecApro International S. A.

（续）

序号	算法名称	原有算法	申 请 人
7	HPC	—	Rich Schroeppel
8	LOKI97	LOKI89，91	Lawrie Brown，Josef Pieprzyk，Jemnifer Seberry
9	MAGENTA	—	Deutsche Telekom AG
10	MARS	—	IBM
11	RC6	RC5	RSA Laboratories
12	RIJNDAEL	—	Joan Daemen，Vincent Rijmen
13	SAFER +	SAFER	Cylink Corporation
14	SERPENT	—	Ross Anderson，Eli Biham，Lars Knudsen
15	TWOFISH	—	B. Schreier，J. Kelsey，D. Whiting，D. Wagner，C. Hall，N. Ferguson

这 15 个候选算法的基本原理各有不同，大部分是基于 Feistel 网络的，也有一些是基于扩展的 Feistel 网络和 SP 网络，如表 3-2 所示。

表 3-2 15 个 AES 候选算法的基本原理

序号	算法名称	基本原理					
		Feistel/网络（轮数）	扩展 Feistel 网络		SP 网络/（轮数）	其他	
			轮数	圈数		类型	轮数
1	CAST-256		48	12			
2	CRYPTON				12		
3	DEAL	6，6，8					
4	DFC	8					
5	E2	12					
6	FROG					Key Interp	8
7	HPC					Omni	8
8	LOKI97	16					
9	MAGENTA	6，6，8					
10	MARS		32	16			
11	RC6		20	10			
12	RIJNDAEL				10，12，14		
13	SAFER +				8，12，16		
14	SERPENT				32		
15	TWOFISH	16					

在第二次 AES 候选会议上，NIST 确定了 5 个候选算法，它们是 MARS、RC6、Twofish、Serpent 和 Rijndael。下面分别介绍这 5 个第二轮获胜的算法。

3.2 MARS 密码算法

MARS 采用非平衡的 Feistel 网络，所有内部操作均以 32bit 字为单位。它支持 128bit 数

据块加密及可变长密钥，密钥长度从128bit直至超过400bit。MARS拥有比三重DES更高的安全性，同时也拥有比DES快得多的运行速度。另外，可以非常简洁地用软件或硬件方式实现MARS，能够轻松地将其应用于一些智能卡及有限资源的环境中。以MARS的C实现为例，在一台200MHz的Pentium-Pro上，它的运行速度可以达到65Mbit/s，而在一台200MHz的Power-PC上，它的运行速度可以达到85 Mbit/s，经过完全优化后，它可以达到100 Mbit/s的运行速度。在硬件方面，MARS可以达到10倍的加速效果。

3.2.1 MARS算法描述

MARS的输入、输出均采用4个32bit的字，算法本身也是面向字的，它所有的内部操作也是基于32bit字的。该算法的整体结构如图3-1所示，包括三个阶段：

第一阶段（前向混合）——包括添加密钥字到数据字上以及8轮基于S盒的无需密钥的Type-3 Feistel混合。该过程提供的快速混合及密钥雪崩效应有效地阻止了选择明文攻击，加大了线性攻击和差分攻击中逐轮对密码核心进行的攻击。

第二阶段（密码核心）——包括16轮在密钥控制下的Type-3 Feistel密码变换。为了保证加密和解密具有同样的强度，前8轮以前向模式变换，而后8轮以后向模式变换。

第三阶段（后向混合）——该过程同样进行了快速混合，具有很强的雪崩效应，该阶段主要是为了抵抗选择密文攻击。它可以看作是阶段一的逆运算，也包括8轮相同的Type-3 Feistel混合，只是采用的是后向模式，接着从数据字中减去密钥字。

在后续描述中，以D［］表示32bit的数据字数组，数组长度为4。开始在数组D中保存的是明文字，加密结束后它包含的是密文字；K［］是扩展密钥数组，含有40个32bit的密钥字；S［］为S盒，包含512个32bit字，将该数组中的前256项记为S0，后256项记为S1。以上数组的起始标号均为0。

（1）阶段一：前向混合

首先将每一个数据字与密钥字相加，然后施行8轮无密钥的Type-3 Feistel混合，同时还要进行一些额外的混合运算。在每一轮中，使用1个数据字（源字）去修改其他3个数据字（目标字），具体做法是将源字的4个字节作为S盒（S0和S1）的索引，得到相应的数据项，然后将其与目标字相加或异或。首先，将源字的4个字节由低到高分别标记为b0、b1、b2、b3，将b0和b2作为S0的索引，而把b1和b3作为S1的索引，然后将S0［b0］与第1个目标字异或，再与S1［b1］相加；S0［b2］与第2个目标字相加；S1［b3］与第3个目标字异或。最后将源字右移24位，完成一轮操作。

在下一轮，对这4个字进行移位操作如下：当前的第1个目标字成为新的源字，第2个目标字成为第1个目标字，第3个目标字成为第2个目标字，而当前的源字成为第3个目标字。之后进行新的轮操作。

此外，每隔4轮就将1个目标字与源字相加，作为新的源字，例如，在第1轮和第5轮之后，将第3个目标字加到源字上，而第2轮和第6轮后，则将第1个目标字加到源字上。这样做的目的是消除混合的对称性，以消除差分攻击，获得较好的雪崩效应。

该阶段如图3-2所示。其中，⊕表示两个字按位异或，⊞表示模2^{32}加，S_0、S_1分别表示两个8×32的S盒，8 ⋘ 表示循环左移8位。

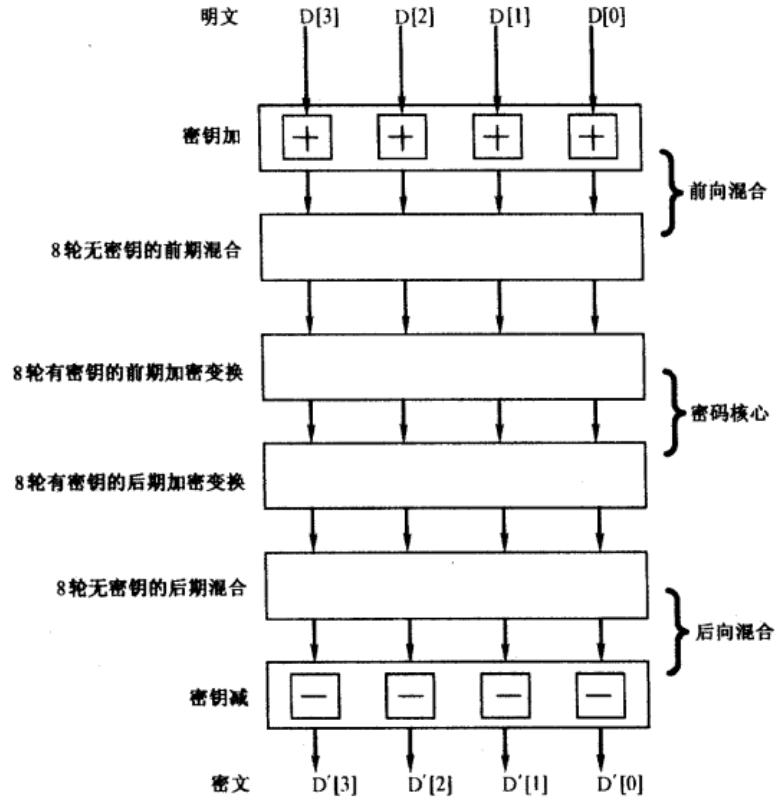


图 3-1　MARS 算法整体结构

（2）阶段二：加密变换

MARS 加密阶段采用的仍是 Type-3 Feistel 网络，共有 16 轮变换，每一轮都进行一次扩展函数（E—函数）变换，该函数输入 1 个数据字，输出为 3 个数据字，中间要经过乘法、数据移位以及 S 盒查询等操作。该 Feistel 网络的结构如图 3-3 所示。

在加密的前 8 轮，E 函数的第 1 个和第 2 个输出分别与前两个目标字相加，而第 3 个输出与第 3 个目标字异或；而在后 8 轮，则是将其第 1 个和第 2 个输出分别与第 3 和第 2 个目标字异或，第 3 个输出与第 1 个目标字异或，之所以这样做是为了使该加密算法能够有效地抵抗选择密文攻击。

E 函数的过程如图 3-4 所示，在 E 函数中有 3 个中间变量，分别表示为 L、M 和 R，也称为 E 函数的 3 条线。该函数的各过程分别标示为 1，2，…，8：

1）将源字循环左移 13 位作为 R 值；

2）源字与第 1 个密钥相加，其和作为 M 值；

3）以 M 的低 9 位作为索引，在 S 盒中查询相应的表项（该 S 盒共有 512 个表项，由前期混合阶段得到的 S1 和 S2 构成），并令 L 等于相应表项中的值；

4）将 R 与第 2 个密钥字相乘，并将 R 值循环左移 5 位；

5）取 L 为 R 与 L 异或的值；

6）取 R 的最低 5 位构成的值为移位次数（LT），该值介于 0 与 31 之间，将 M 值循环左移 LT 位；

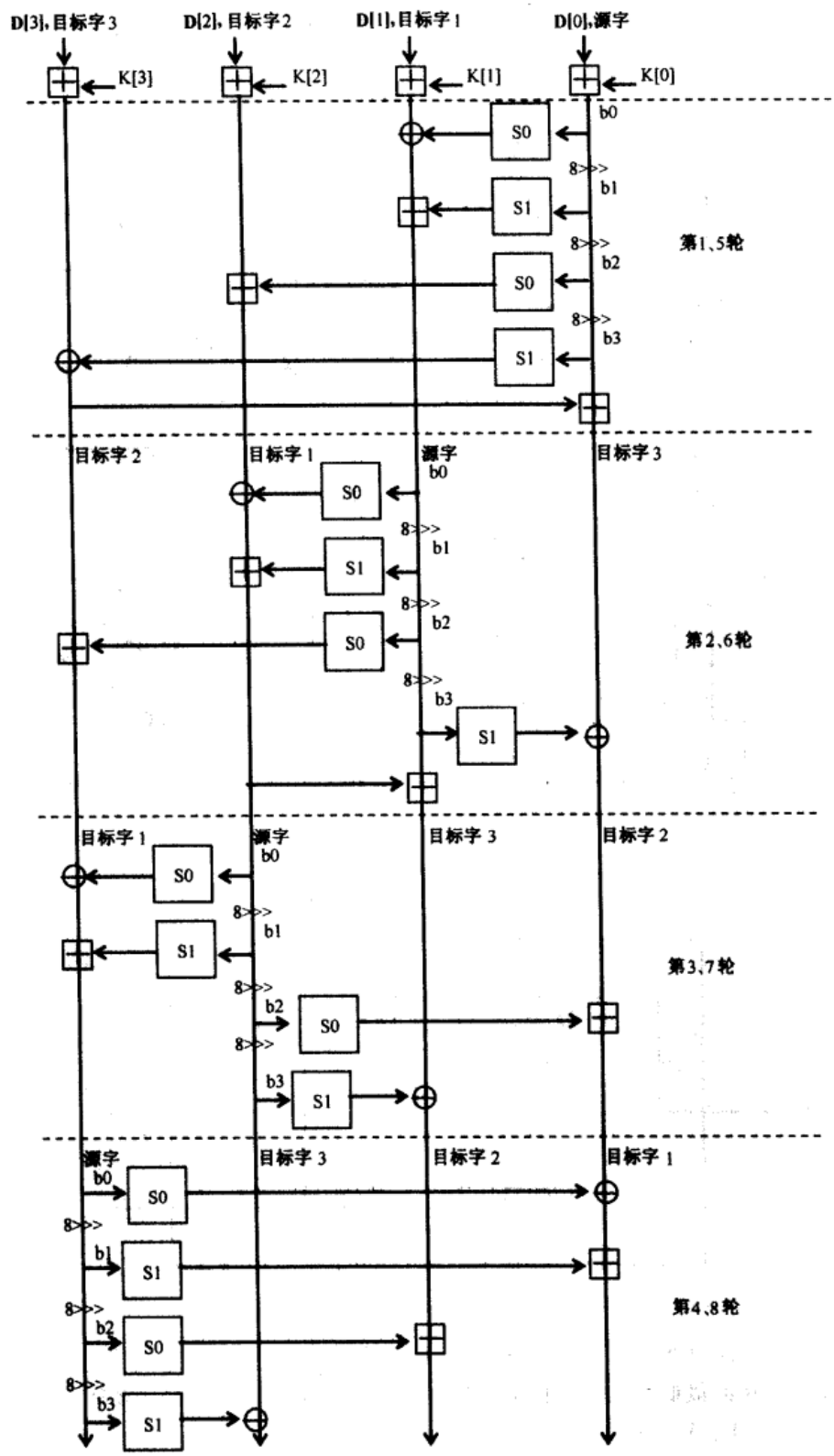

D[3],目标字3
D[2],目标字2
D[1],目标字1
D[0],源字
K[3]
K[2]
K[1]
K[0]
S0
S1
b0
b1
b2
b3
8>>
第1、5轮
目标字2
目标字1
源字
目标字3
第2、6轮
目标字1
源字
目标字3
目标字2
第3、7轮
源字
目标字3
目标字2
目标字1
第4、8轮

图 3-2　前向混合阶段

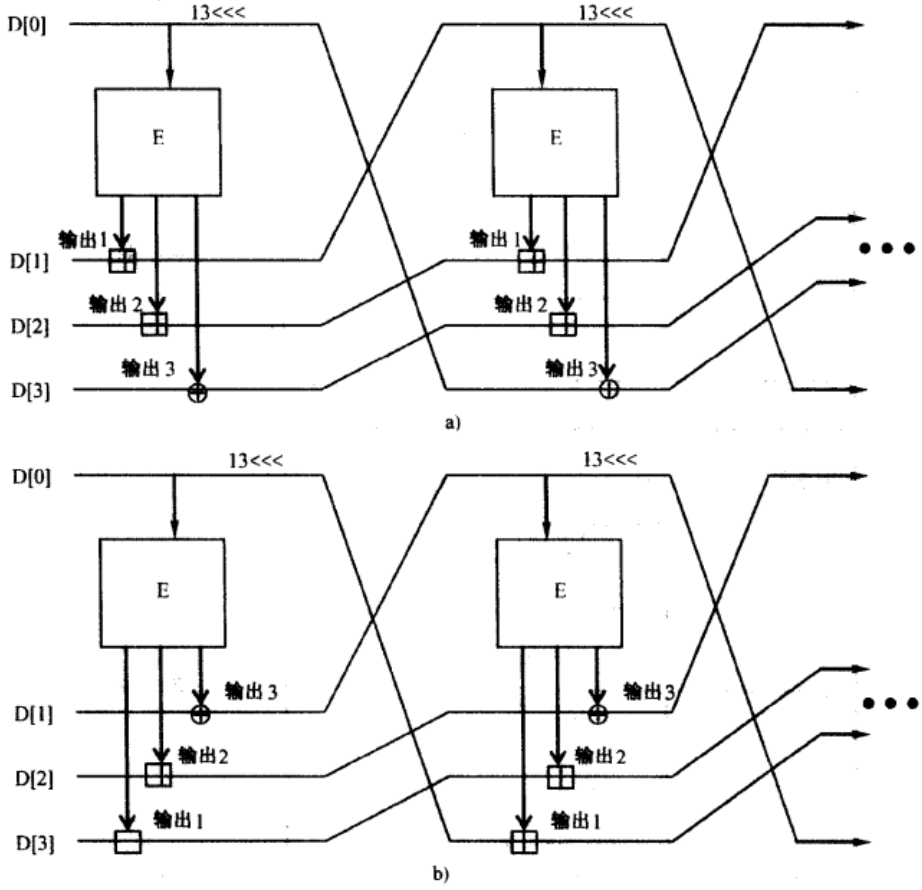


图 3-3　Feistel 网络

a）前向加密模式　b）后向加密模式

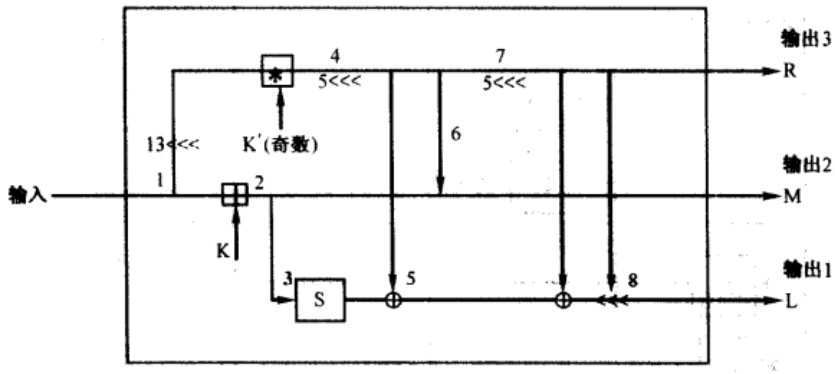


图 3-4　E 函数

7）将 R 再循环左移 5 位（或多于 5 位），L 与其异或；

8）最后仍将 R 的最低 5 位作为移位次数 LT，将 L 循环左移 LT 位。这样 E 函数的第 1、2、3 个输出分别为 L、M、R。

这样设计的目的主要是为了消除数据间的依赖性，减少运算中各数据位变换的规律性。

（3）阶段三：后向混合

后向混合阶段相当于前向混合阶段的逆过程，只是数据字的使用顺序不同，如果将前向过程的输出逆序作为后向阶段的输入，那么这两个过程将互相抵消。该阶段的流程如图 3-5 所示。

与前向阶段一样，该阶段在每一轮也是用 1 个源字去修改其余 3 个目标字，分别用 b0、b1、b2、b3 表示源字的 4 个字节，与前向阶段的区别主要体现在以下方面：

1）使用 b0、b2 作为 S 盒 S1 的索引，而用 b1、b3 作为 S0 盒的索引；

2）第 1 目标字与 S1［b0］异或，从第 2 目标字中减去 S0［b3］，从第 3 目标字中减去 S1［b2］，然后再将 S0［b1］与第 3 目标字异或；

3）每一轮结束后，将源字循环左移 24 位；

4）在每 4 轮之前，都要从源字中减去 1 个目标字，具体做法为：在第 4、8 轮之前，从源字中减去第 1 目标字，在第 3、7 轮之前，从源字中减去第 3 目标字。

MARS 的解密过程是加密过程的逆运算。

（4）MARS 加/解密过程的伪代码表示

下面以伪代码的形式给出 MARS 加/解密过程。

1）E 函数（输入：in，key1，key2；中间变量：L，M，R）。

```
[1] M = in + key1                          //先与第 1 个密钥字相加
[2] R = (in ⋘ 13) × key2                   //与第 2 个密钥字相乘
[3] i = M 的最低 9 位
[4] L = S[i]                               //查找 S 盒
[5] R = R ⋘ 5
[6] r = R 的最低 5 位                      //该值规定了移位量
[7] M = M ⋘ r
[8] L = L⊕R
[9] R = R ⋘ 5
[10] L = L⊕R
[11] r = R 的最低位
[12] L = L ⋘ r
[13] 输出(L,M,R)
```

2)加密(输入 D[ ],K[ ])。

- 前向混合。

```
[1]for i = 0 to 3 do
        D[i] = D[i] + K[i]                 //将子密钥与数据字相加
[2]//8 轮前向混合
[3]for i = 0 to 7 do                       //采用 D[0]修改 D[1],D[2],D[3]
[4]     // 4 次 S 盒查找
[5]     D[1] = D[1] ⊕S0[ D[0]的低字节]
[6]     D[1] = D[1] + S1[ D[0]的次低字节]
[7]     D[2] = D[2] + S0[ D[0]的次高字节]
```

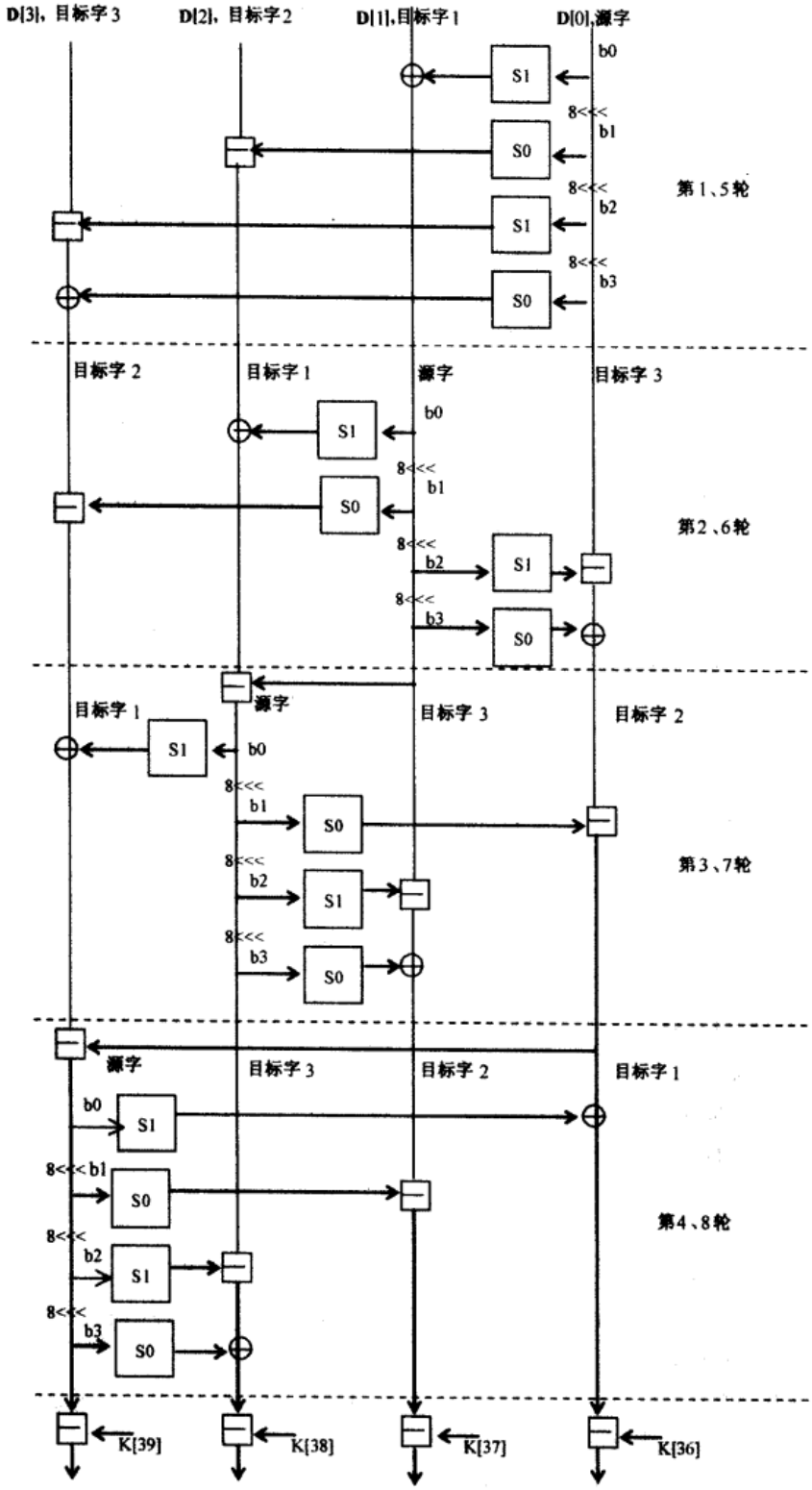

D[3], 目标字3
D[2], 目标字2
D[1], 目标字1
D[0], 源字
S1
S0
b0
b1
b2
b3
8<<<
第1、5轮
目标字2
目标字1
源字
目标字3
第2、6轮
第3、7轮
第4、8轮
K[39]
K[38]
K[37]
K[36]

图 3-5　后向混合阶段

```
[8]    D[3] = D[3] ⊕S1[D[0]的高字节]
[9]    //将源字右移
[10]    D[0] = D[0] ⋙24
[11]    //额外的混合操作
[12]    if i = 0 or 4 then
[13]      D[0] = D[0] + D[3]          //将 D[3]加到源字上
[14]    if i = 1 or 5 then
[15]      D[0] = D[0] + D[1]          //将 D[1]加到源字上
[16]    (D[3],D[2],D[1],D[0])←(D[0],D[3],D[2],D[1])
[17]end-for
```

- 加密变换(16 轮加密变换)。

```
[18]for i = 0 to 15 do
[19]    (out1,out2,out3) = E - 函数(D[0],K[2i+4],K[2i+5])
[20]    D[0] = D[0] ⋘13
[21]    D[2] = D[2] + out2
[22]    if i < 8 then                  //前向模式的 8 轮
[23]      D[1] = D[1] + out1
[24]      D[3] = D[3] ⊕out3
[25]    else                           //后向模式的 8 轮
[26]      D[3] = D[3] + out1
[27]      D[1] = D[1] ⊕out3
[28]    end-if
[29]    (D[3],D[2],D[1],D[0]) ←(D[0],D[3],D[2],D[1])
[30]end-for
```

- 后向混合 (8 轮向后混合)。

```
[31] for i = 0 to 7 do
[32]    if i = 2 or 6 then             //额外的混合运算
[33]      D[0] = D[0] - D[3]           //从源字中减去 D[3]
[34]    if i = 3 or 7 then
[35]      D[0] = D[0] - D[1] //从源字中减去 D[1]
[36] // 4 次 S 盒查找
[37] D[1] = D[1]⊕S1[D[0]的低字节]
[38] D[2] = D[2] - S0[D[0]的高字节]
[39] D[3] = D[3] - S1[D[0]的第 3 个字节]
[40] D[4] = D[3]⊕S0[D[0]的第 2 个字节]
[41] D[0] = D[0] ⋘24                  //源字左移
[42] (D[3],D[2],D[1],D[0])←(D[0],D[3],D[2],D[1])
```

```
[43] end-for
[44] for i = 0 to 3 do //从数据字中减去子密钥
[45] D[i] = D[i] - K[36 + i]
```

3)解密(输入 D[ ],K[ ])。

- 前向混合。

```
[1] for i = 0 to 3 do
[2]   D[i] = D[i] + K[36 + i]                    //首先将子密钥与数据字相加
[3] for i = 7 downto o do                        //8 轮前向混合
[4]   (D[3],D[2],D[1],D[0])←(D[2],D[1],D[0],D[3]) //将 D[ ]左移
[5]   (D[0]) = D[0] ⋙ 24                         //源字右移
[6]   //4 次 S 盒查找
[7]   D[3] = D[3]⊕S0[D[0]的第二字节]
[8]   D[3] = D[3] + S1[D[0]的第三字节]
[9]   D[2] = D[2] + S0[D[0]的高字节]
[10]  D[1] = D[1]⊕S1[D[0]的低字节]
[11]  //额外的混合操作
[12]if i = 2 or 6 then
[13]  D[0] = D[0] + D[3]                         //将 D[3]加到源字上
[14]if i = 3 or 7 then
[15]  D[0] = D[0] + D[1]                         //将 D[1]加到源字上
[16]end-for
```

- 解密变换(16 轮解密变换)。

```
[17]for i = 15 downto 0 do
[18]  (D[3],D[2],D[1],D[0]) ←(D[2],D[1],D[0],D[3]) //D[ ]左移
[19]  D[0] = D[0] ⋙ 13
[20]  (out1,out2,out3) = E - 函数(D[0],K[2i + 4],K[2i + 5])
[21]  D[2] = D[2] - out2
[22]  if i < 8 then                              //后 8 轮,前向模式
[23]    D[1] = D[1] - out1
[24]    D[3] = D[3]⊕out3
[25]  else
[26]    D[3] = D[3] - out1
[27]    D[1] = D[1]⊕out3
[28]  end-if
[29]end-for
```

• 后向混合(8 轮向后混合)。

```
[30] for i = 7 downto 0 do
[31]  (D[3],D[2],D[1],D[0])←(D[2],D[1],D[0],D[3]) //D[ ]左移
[32]  //额外的混合操作
[33]  if i = 0 or 4 then
[34]    D[0] = D[0] - D[3]              //从源字中减去 D[3]
[35]  if i = 1 or 5 then
[36]    D[0] = D[0] - D[1]              //从源字中减去 D[1]
[37]    D[0] = D[0] ⋘ 24                //源字左移
[38]  //4 次 S 盒查找
[39]  D[3] = D[3]⊕S1[D[0]的高字节]
[40]  D[2] = D[2] - S0[D[0]的第三个字节]
[41]  D[1] = D[1] - S1[D[0]的第二个字节]
[42]  D[1] = D[1]⊕S0[D[0]的低字节]
[43]  end-for
[44]for i = 0 to 3 do                   //从数据字中减去子密钥
[45]  D[i] = D[i] - K[i]
[46]end-for
```

3.2.2 MARS 安全性分析

在分析 MARS 安全性之前，先介绍一些有关安全攻击的术语。

数据复杂性（Data Complexity）：攻击的数据复杂性是指为使一次攻击成功，攻击者所需的（明文，密文）对的数目。

工作负载（Work Load）：指完成一次攻击所需的操作数。一般而言，一次攻击的工作负载至少等于其数据复杂性，有时还会更大。例如穷举攻击只需 2～3 个（明文，密文）对就可以了，其数据复杂性很小，但工作负载却相当大，为 $2^{\text{密钥长度}-1}$。

密钥概率（Key Probability）：有些攻击只能在密钥具有某些特定属性时才能进行，其密钥概率就是一个随机密钥具有该属性的概率。在计算密钥概率时，均假设密钥是独立、随机选取的。

安全级别（Security Level）：一个加密算法在某次（类）攻击时的安全级别是该攻击所需的工作负载与密钥概率的比值。比如，如果某攻击的工作负载为 2^{20}，密钥概率为 2^{-30}，那么该密码算法针对此攻击的安全级别为 2^{50}。

据 IBM MARS 工作组证明，当密钥长度为 $n$bit（$n \leqslant 256$）时的算法安全级别为 2^n，但当 $n > 256$ 时，安全级别不会超过 2^{256}；针对 MARS 的线性攻击或差分攻击的数据复杂性都大于 2^{128}，因此当分组长度为 128bit 时，这些攻击是不可能成功实施的。

相关技术文献给出了以上结论的详细证明，在此略。

3.2.3 MARS 性能测试

IBM 公司在不同平台下对 MARS 分别采用软、硬件实现，并给出了测试结果。

（1）软件实现

1）C 实现

在时钟速率为 200MHz，运行速度为 65～85Mbit/s 的机器上，MARS 的 C 语言实现性能如表 3-3 所示。

表 3-3　MARS 的 C 语言实现时的运行时间

	Pentium-Pro Borland C + +5.0	Pentium-Pro DJGPP（+pgcc101）	PowerPC 604e CSet + +3.1.1
加密	28Mbit/s	65Mbit/s	85Mbit/s
解密	28Mbit/s	65Mbit/s	85Mbit/s
建立密钥	8500 时钟周期	3850 时钟周期	1650 时钟周期
改变密钥	8500 时钟周期	3850 时间周期	1650 时钟周期

2）Java 实现。

在同样的平台上对 MARS 的 Java 实现性能如表 3-4 所示。

表 3-4　MARS 的 Java 语言实现时的运行时间

	Pentium-Pro	PowerPC 604e
加密	17.2Mbit/s	23.0Mbit/s
解密	17.6Mbit/s	23.2Mbit/s
建立密钥	5000 时钟周期	5600 时钟周期
改变密钥	5000 时钟周期	5600 时钟周期

（2）在 8 位处理器上的实现

在 8 位处理器上的 MARS 加、解密及密钥建立过程的运行速率如表 3-5 所示。

表 3-5　MARS 在 8 位处理器上的运行速率

运算	操作次数	时钟周期/操作	总时钟周期
乘法	16	73	1168
移位（移位次数与数据相关）	32	34	1088
固定移位	48	12	576
8-bit S [ ]	64	8	512
9-bit S [ ]	16	12	192
其他	184	8	1472
加、解密总过程			5008
移位（移位次数与数据相关）	16	34	544
固定移位	480	12	5760
9-bit S [ ]	296	12	3552
其他	740	8	5920
密钥建立过程			15776

（3）硬件实现

MARS 算法很适于用硬件实现。若只使用一个乘法器，所需单元估计为 70000 个，运行速率为 640Mbit/s，而且这些单元可集成到所有芯片上，也适用于智能卡。因此 MARS 算法

可灵活地运用在多种应用中。

对于采用 Pipelining 的操作（如 ECB 模式、CBC 模式），硬件执行速度更快，加/解密速率可达到 4Gbit/s 甚至是 8Gbit/s，所需单元数大概为 393000 个。

3.3 RC6 密码算法

RC6 是 RC5 的改进算法，主要表现在用 4 个工作寄存器代替 2 个，并将整数乘法作为一个基本的操作，从而增加了每轮循环的数据扩散度，提高了安全性，减少了轮数，同时提高了处理量。这也是为了满足 AES 的要求而提出的。

3.3.1 RC6 算法描述

与 RC5 一样，RC6 算法是完全参数化的，它可以表示成 RC6 - w/r/b 的形式，其中 w 表示字长，r 表示轮数，b 表示以字节计算的密钥长度。RC6 算法的密钥长度可为 16、24 或 32bit。AES 提案主要是针对 w = 32，r = 20 的情况，对于其他情况可以表示为 RC6 - w/r。

RC6 - w/r/b 在 4 个 w 位长的字上采用以下 6 种基本的操作：

a + b：整数模 2^w 加法；

a - b：整数模 2^w 减法；

a⊕b：按位异或；

a × b：整数模 2^w 乘法；

a < < < b：将 a 左移，移位数为 b 的最低 lgw 位（lgw 表示以 2 为底的 w 的对数）；

a > > > b：将 a 右移，移位数为 b 的最低 lgw 位。

在 RC6 的每一轮中都采用一半数据去修改另一半数据，与 RC5 算法一样，称其为“half-round”模式。

（1）密钥建立方案

在 RC6-w/r/b 中，由用户提供一个 b 字节的密钥，0≤b≤255，由此可得到 2r + 4 个字，每个字为 w 位，并存在数组 S [0，…，2r + 3] 中，该密钥数组在加密、解密过程中都会用到。

首先用户提供的密钥被输入 c 个数组中，数组元素为 w 位的字，将其表示为 L [0]，…，L [c - 1]，若位数不够则用 0 填充。密钥扩展中还会使用两个常数，P_{32} = B7E15163，Q_{32} = 9E3779B9，它们分别是由 e - 2 和 ϕ - 1 的二进制扩展中得到的，其中 e 是自然对数的底，ϕ 为黄金分割率。具体过程为：

输入：L [0，…，c - 1]，轮数 r

输出：S [0，…，2r + 3] 即为扩展子密钥

S [0] = Pw

for i = 1 to 2r + 3 do

  S [i] = S [i - 1] + Qw

A = B = i = j = 0

v = 3 * max {c，2r + 4}

for s = 1 to v do

```
    {
     A=S[i] = (S[i] +A+B) <<<3
     B=L[j] = (L[j] +A+B) <<< (A+B)
     i= (i+1) mod (2r+4)
    }
```

(2) 加密

在 RC6 中使用了 4 个 w 位的寄存器，最初寄存器中存放的是明文数据，加密结束后，其内容为密文结果。在存储方式上，原文或密文的第一个字节存放在 A 寄存器的最低字节内，而其最后一个字节则存放在 D 寄存器的最高字节内。(A, B, C, D) = (B, C, D, A) 表示将右边寄存器的值平行赋给左边寄存器。

加密过程可用伪代码表示为：

```
B=B+S[0]
D=D+S[1]
for i=1 to r do
    {
     t= (B× (2B+1)) <<<lgw
     u= (D× (2D+1)) <<<lgw
     A= ((A⊕t) <<<u) +S[2i]
     C= ((C⊕u) <<<t) +S[2i+1]
      (A, B, C, D) = (B, C, D, A)
    }
A=A+S[2r+2]
C=C+S[2r+3]
```

输出 A、B、C、D 即为密文。该过程如图 3-6 所示。

(3) 解密

解密过程是加密过程的逆运算，密文也放在 A、B、C、D 寄存器中，r 为轮数，各轮的子密钥 S[0, …, 2r+3] 为 w 位，经解密后的明文也放在 4 个寄存器中。

解密过程的伪代码表示为：

```
C=C-S[2r+3]
A=A-S[2r+2]
for i=r downto 1 do
    {
      (A, B, C, D) = (D, A, B, C)
     u= (D× (2D+1)) <<<lgw
     t= (B× (2B+1)) <<<lgw
     C= ((C-S[2i+1]) >>>t) ⊕u
     A= ((A-S[2i]) >>>u) ⊕t
    }
D=D-S[1]
```

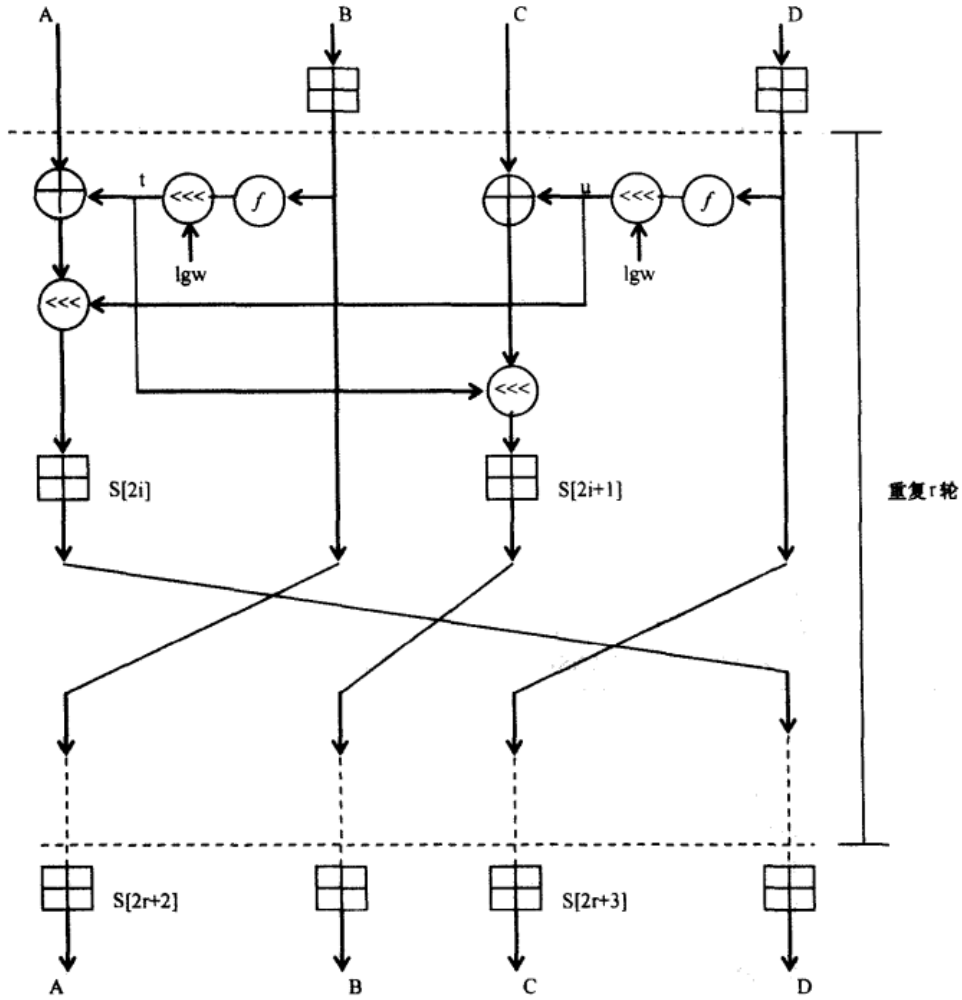


图 3-6　RC6 加密

B = B - S [0]

输出 A、B、C、D 即为明文。

3.3.2　RC6 安全性分析

Rivest 等人对 RC6 算法的安全性进行了定性证明，得出以下结论：

1）针对 RC6 最有效的攻击是穷举攻击，其工作负载为 $\min\{2^{8b}, 2^{1408}\}$。

2）对 RC6 进行差分攻击或线性攻击（及其各种改进攻击方案）所需的数据量均超过可用的数据量。

3）不能够证明存在弱密钥。

Knudsen 及 Meier 等还对 RC6 进行了相关性分析（Correlation Attack），结果表明不能对 20 轮的完整 RC6 算法实施有效的攻击。

3.3.3　RC6 性能测试

由上所述，RC6 算法是非常简单的，算法设计者就其加/解密时间、密钥建立时间等进

行了测试。

（1）软件实现

1）使用 ANSI C、Java 和汇编语言的加/解密速度如表 3-6 所示。

表 3-6　RC6 在 ANSI C、Java、汇编语言中实现时的加/解密速度

	方案	时钟周期/明密文块	明密文块/s (200MHz)	MB/s (200MHz)
ANSI C	RC6 加密	616	325000	5.19
ANSI C	RC6 解密	566	353000	5.65
Java（JDK）	RC6 加密	16200	12300	0.197
Java（JDK）	RC6 解密	16500	12100	0.194
Java（JIT）	RC6 加密	1010	197000	3.15
Java（JIT）	RC6 解密	955	209000	3.35
汇编	RC6 加密	254	787000	12.6
汇编	RC6 解密	254	788000	12.6

2）用 ANSI C、Java 的密钥建立如表 3-7 所示。

表 3-7　RC6 在 ANSI C、Java 实现时的密钥建立速度

	方案	时钟周期数	μs/密钥 (200MHz)	建立的密钥数/s (200MHz)
ANSI C	RC6-32/20/16	4710	23.5	42500
Java（JDK）	RC6-32/20/16	107000	537	1860
Java（JIT）	RC6-32/20/16	14300	71.4	14000
ANSI C	RC6-32/20/24	4710	23.6	42400
Java（JDK）	RC6-32/20/24	108000	542	1840
Java（JIT）	RC6-32/20/24	14300	71.5	14000
ANSI C	RC6-32/20/32	4720	23.6	42400
Java（JDK）	RC6-32/20/32	110000	548	1820
Java（JIT）	RC6-32/20/32	15000	75.1	13300

（2）RC6 在 8 位处理器上的性能

以 Intel 的 MCS-51 微处理器系列为例，在一轮 RC6 加/解密运算中包含 6 次加法、2 次异或、2 次乘方运算、2 次左移（移位量固定为 5 位）以及 2 次变长移位运算。由此可得到一轮所需的指令周期，如表 3-8 所示。

表 3-8　一轮 RC6 在 8 位处理器上实现时所需的指令周期数

运算	指令	时钟周期/操作	总时钟周期
加	4ADDC	4	4×6=24
异或	4XRL	4	4×2=8
乘方	6MUL，11ADC	35	35×2=70
循环左移 5 位	12RRC	12	12×2=24
循环左移 r 位	8RRC（RLC） 8JB	24	24×2=48
总计			174

至于 RC6 的密钥建立，主要集中在最后一个循环中，若 $b=16$，24，32，$r=20$，则循环次数为 $3\times\max\{20\times2+4,\ b/4\}=132$，与 b 无关。与加/解密过程的分析类似，在一次循环中所需的指令数为 52，因此对密钥建立过程来说，所需的指令周期为 $(52\times132)\times4=27456$，所需时间为 27ms。

（3）硬件实现

RC6 算法所涉及的原子操作，如加、减、乘、异或以及移位等都是现代微处理器所支持的，因此在硬件实现上可以利用已有的电路模块，采用门电路技术实现 RC6 算法。但对大多数应用来说，用软件实现是较好的选择，而且采用门电路技术对 RC6 算法的硬件实现性能要比微处理器实现差，不适合智能卡实现。

3.4 Twofish 密码算法

Twofish 算法的分组长度为 128bit，密钥长度最多为 256bit。它采用了双射 Feistel 网络变换，包括 8 个 S 盒置换、一个基于 MDS（Maximum Distance Separable）矩阵的变换、一个伪 Hadamard 变换等，同时对密钥表也进行了精心地设计。

3.4.1 Twofish 算法描述

Twofish 的总体结构是一个 16 轮的 Feistel 网络，对输入和输出还要进行白化处理。该算法如图 3-7 所示。

首先，128 位原文被分成 4 个 32 位字，这 4 个数据字分别与 4 个 32 位密钥字异或，即对输入进行白化处理。之后进行 16 轮循环，在每一轮循环中，前两个字分别作为两个 g 函数的输入，其中一个字先要左移 8 位，而每个 g 函数包含 4 个与密钥相关的 S 盒，4 个 S 盒的输出先要基于 MDS 矩阵进行混合操作，然后将 2 个 g 函数的输出进行 PHT 混合，再与两个密钥相加，至此完成了一轮 F 函数操作。F 函数的第 1 个输出与第 3 个输入字异或，然后右移 1 位，而第 4 个输入字左移 1 位后与 F 函数的第 2 个输出异或。最后将左右两部分互换，完成一轮操作，即第 3、4 个输入字作为下一轮的第 1、2 个输入字，而本轮的第 1、2 个输入字分别作为下一轮的第 3、4 个输入字。完成 16 轮迭代后（其中第 16 轮不进行左右互换）再进行输出白化处理，从而得到输出密文。该过程可表示为：

1）初始白化。

$$R_{0,i}=P_i\oplus K_i\quad i=0,\ \cdots,\ 3$$

2）16 轮加密。

设 $R_{r,0}$、$R_{r,1}$、$R_{r,2}$、$R_{r,3}$为第 r 轮中的 4 个输入，输出为 $R_{r+1,0}$、$R_{r+1,1}$、$R_{r+1,2}$、$R_{r+1,3}$，其过程为：

$$R_{r+1,0}=\mathrm{ROR}((g(R_{r,0})+g(\mathrm{ROL}(R_{r,1},8)+K_{2r+8})\oplus R_{r,2},1)$$

$$R_{r+1,1}=(g(R_{r,0})+2g(\mathrm{ROL}(R_{r,1},8)+K_{2r+9})\oplus\mathrm{ROL}(R_{r,3},1)$$

$$R_{r+1,2}=R_{r,0}$$

$$R_{r+1,3}=R_{r,1}$$

其中，ROR（X，m）、ROL（X，m）分别表示将 X 右移或左移 m 位，+表示模 2^{32}加。

3）输出白化。

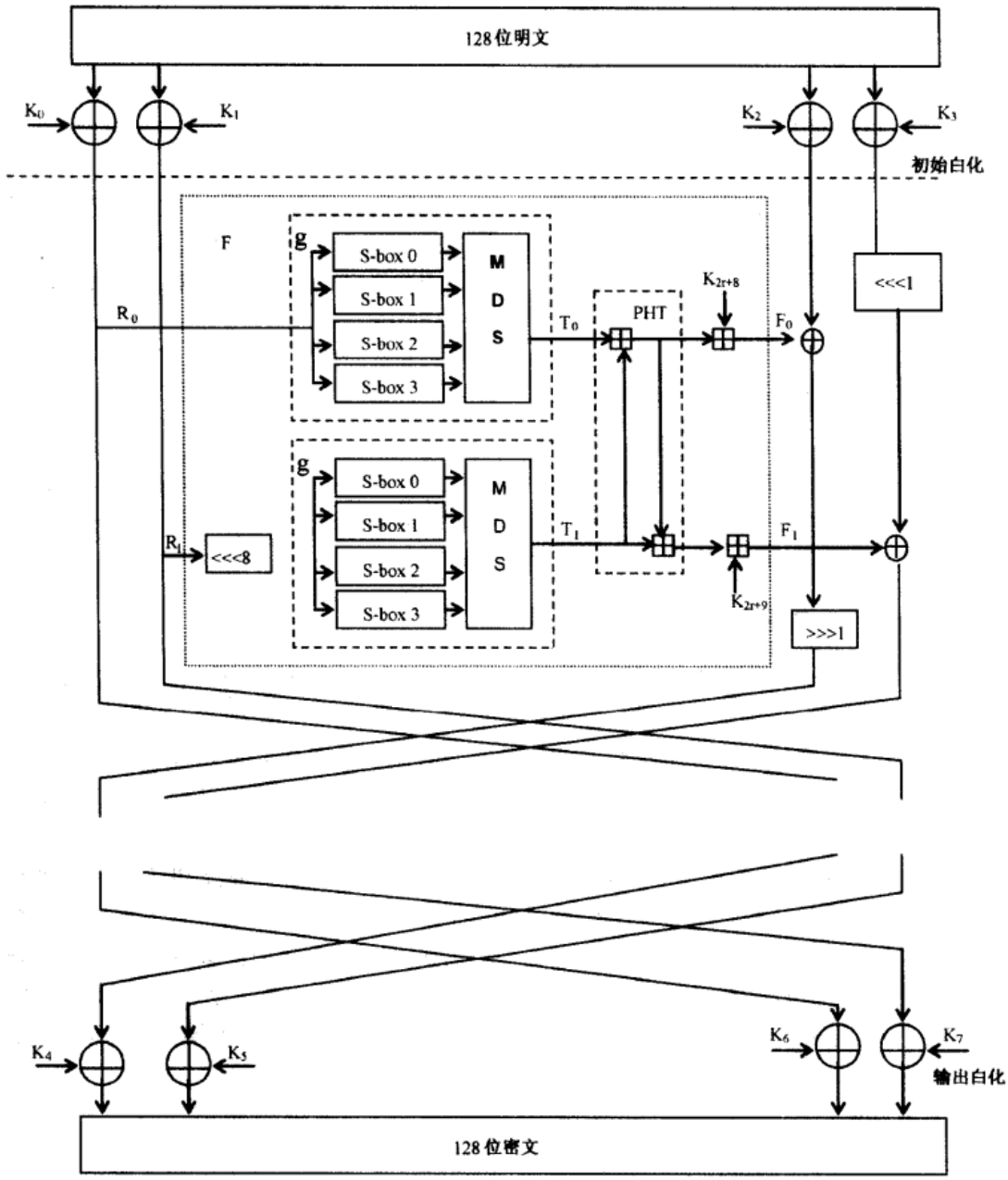


图 3-7　Twofish 算法流程

设第 16 轮输出（不经过左右对换）分别为 $R_{17,0}$、$R_{17,1}$、$R_{17,2}$、$R_{17,3}$，依次与密钥 K_4、K_5、K_6、K_7 异或，得到密文。

下面详细看一下每一轮的迭代过程。

（1）F 函数

F 函数是一个置换过程，当前轮数决定了所用的子密钥。若输入记为 $R0$、$R1$，则 F 函数的过程可表示为：

$T_0 = g\ (R_0)$

$T_1 = g$（ROL（R_1，8））

F_0 =（$T_0 + T_1 + K_{2r+8}$）mod2^{32}

$F1$ =（$T_0 + 2T_1 + K_{2r+9}$）mod2^{32}

其中，F_0、F_1 即为 F 函数的输出结果。

（2）g 函数

g 函数是整个算法的核心。它将输入的 32 位分为 4 个字节，分别送入 4 个 S 盒，每个 S 盒的输出为 8 位，这 4 个 S 盒的输出可表示为有限域 GF（2^8）上长度为 4 的向量，并与 4 × 4 的 MDS 矩阵相乘，乘积向量所代表的 32 位字即为 g 函数的输出。设 X 为 g 函数的 32 位输入，按高位在前，可将它表示成 4 个字节，即 X =（x_3，x_2，x_1，x_0），则 g 函数的处理过程可描述为：

$$x_i = [X/2^{8i}] \bmod 2^8 \quad i = 0,\cdots,3$$

$$y_i = S_i[x_i] \quad i = 0,\cdots,3$$

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} \cdot & \cdots & \cdot \\ \vdots & \text{MDS} & \vdots \\ \cdot & \cdots & \cdot \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$Z = \sum_{i=0}^{3} z_i \times 2^{8i}$$

其中，S_i 表示相应的 S 盒，

$$\text{MDS} = \begin{pmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{pmatrix}$$

Z =（$z_3z_2z_1z_0$）是 g 函数的输出。

（3）S 盒

Twofish 中所用的 S 盒是与密钥相关的，也是在产生扩展密钥的过程中得到的，同时在密钥生成过程中还得到了 40 个扩展密钥 K_0，K_1，…，K_{39}。

设 M 为密钥，N 为密钥长度，则 N 可为 128、192、256bit。令 k = N/64，则 M 共有 8k 个字节，分别表示为 m_{8k-1}，…，m_1，m_0，可将其表示成 2k 个 32 位的字，即

$$M_i = \sum_{j=0}^{3} m_{(4i+j)} \cdot 2^{8j}, i = 0,1,\cdots,2k-1$$

然后将这 2k 个字组成 2 个长度为 k 的向量，M_e =（M_0，M_2，…，M_{2k-2}），Mo =（M_1，M_3，…，M_{2k-1}）。由密钥还可以得到另外一个长度为 k 的字向量：

$$\begin{pmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \end{pmatrix} = \begin{pmatrix} & \cdots & \\ \vdots & \mathrm{RS} & \vdots \\ & \cdots & \end{pmatrix} \begin{pmatrix} m_{si} \\ m_{si+1} \\ m_{si+2} \\ m_{si+3} \\ m_{si+4} \\ m_{si+5} \\ m_{si+6} \\ m_{si+7} \end{pmatrix}$$

$$\mathrm{RS} = \begin{pmatrix} 01 & A4 & 55 & 87 & 5A & 58 & DB & 9E \\ A4 & 56 & 82 & F3 & 1E & C6 & 68 & E5 \\ 02 & A1 & FC & C1 & 47 & AE & 3D & 19 \\ A4 & 55 & 87 & 5A & 58 & DB & 9E & 03 \end{pmatrix}$$

取 $S_i = \sum_{j=0}^{3} s_{i,j} 2^{8j} = (s_{i,3} s_{i,2} s_{i,1} s_{i,0})$　$S = (S_0, \cdots, S_{k-2}, S_{k-1})$

g 函数中的 S 盒是由 h 函数决定的，使得 g（X）=h（X，S），下面介绍 h 函数过程。h 函数有两个输入，一个是 32 位字 X，另一个是由 32 位字构成的长度为 k 的列表 L=（L_0，…，L_{k-1}），输出为一个字。流程如图 3-8 所示。

该函数共有 k 步，在每一步中输入的 4 个字节都进行一次 S 盒处理，之后与从表 L 得到的一个字节异或。完成 k 步运算后，再进行一次 S 盒处理，输出的 4 字节与 MDS 矩阵相乘，该 MDS 矩阵即为 g 函数中的 MDS 矩阵。

以低位在后表示，L_i 和 X 可分别表示为：$L_i = l_{i,3} l_{i,2} l_{i,1} l_{i,0}$，$X = x_3 x_2 x_1 x_0$，i=0，…，k−1，j=0，…，3

因此，迭代过程可表示为：$y_{k,j} = x_j$，j=0，…，3。

若 k=4，有 $y_{3,0} = q_1[y_{4,0}] \oplus l_{3,0}$，$y_{3,1} = q_0[y_{4,1}] \oplus l_{3,1}$，$y_{3,2} = q_0[y_{4,2}] \oplus l_{3,2}$，$y_{3,3} = q_1[y_{4,3}] \oplus l_{3,3}$；

若 k≤3，有 $y_{2,0} = q_1[y_{3,0}] \oplus l_{2,0}$，$y_{2,1} = q_1[y_{3,1}] \oplus l_{2,1}$，$y_{2,2} = q_0[y_{3,2}] \oplus l_{2,2}$，$y_{2,3} = q_0[y_{3,3}] \oplus l_{2,3}$。

在所有情况下，都有：

$y_0 = q_1[q_0[q_0[y_{2,0}] \oplus l_{1,0}] \oplus l_{0,0}]$

$y_1 = q_0[q_0[q_1[y_{2,1}] \oplus l_{1,1}] \oplus l_{0,1}]$

$y_2 = q_1[q_1[q_0[y_{2,2}] \oplus l_{1,2}] \oplus l_{0,2}]$

$y_3 = q_0[q_1[q_1[y_{2,3}] \oplus l_{1,3}] \oplus l_{0,3}]$

输出 Z 为：

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} & \cdots & \\ \vdots & \mathrm{MDS} & \vdots \\ & \cdots & \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}, Z = \sum_{i=0}^{3} z_i 2^{8i}$$

在上式中，q_0 和 q_1 是对 8 位值的置换。设输入值为 x，x 为 1B，$x = x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0$，

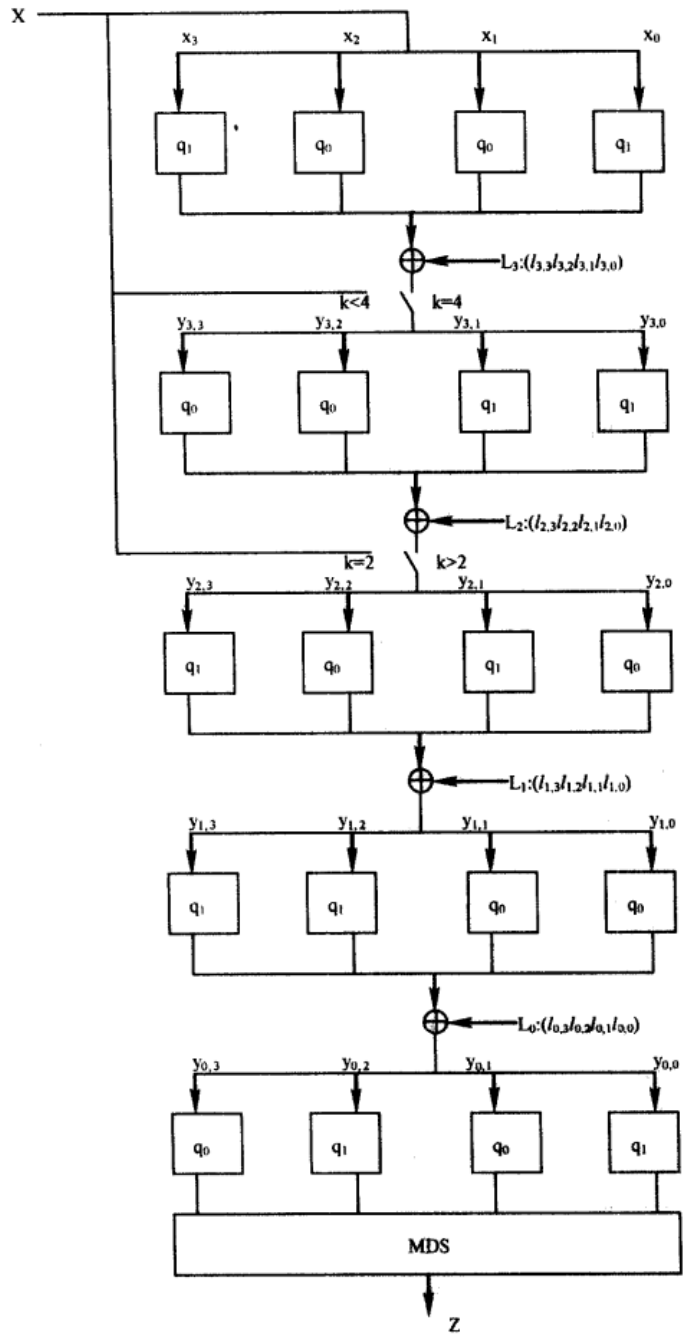


图 3-8　h 函数

x_i 为 1bit，则相应输出值 y 为：

a_0，b_0 = [x/16]，x mod 16，

即　$a_0 = x_7x_6x_5x_4$

$b_0 = x_3x_2x_1x_0$

$a_1 = a_0 \oplus b_0$

$b_1 = a_0 \oplus ROR_4(b_0, 1) \oplus 8a_0 \bmod 16$

a_2，$b_2 = t_0[a_1]$，$t_1[b_1]$

$a_3 = a_2 \oplus b_2$

$b_3 = a_2 \oplus ROR_4(b_2, 1) \oplus 8a_2 \bmod 16$

a_4，$b_4 = t_2[a_3]$，$t_3[b_3]$

$y = 16b_4 + a_4$，即 $y = b_4 a_4$

其中，ROR_4 是对 4 位值的右移操作，对于 q_0 置换有：

t_0 = [8 1 7 D 6 F 3 2 0 B 5 9 E C A 4]

t_1 = [E C B 8 1 2 3 5 F 4 A 6 7 0 9 D]

t_2 = [B A 5 E 6 D 9 0 C 8 F 3 2 4 7 1]

t_3 = [D 7 F 4 1 2 6 E 9 B 3 0 8 5 C A]

对于 q_1 置换，相应的操作为：

t_0 = [2 8 B D F 7 6 E 3 1 9 4 0 A C 5]

t_1 = [1 E 2 B 4 C 3 7 6 D A 5 F 9 0 8]

t_2 = [4 C 7 5 1 6 9 A 0 E D 8 2 B 3 F]

t_3 = [B 9 5 1 C 3 D E 6 4 7 F 2 0 8 A]

t_i（x）的运算方法为：根据 4bit x 所代表的值 m，在表 t_i 中取第 m 列对应的值作为 t_i（x）的输出。

在 S 盒运算中，h 函数的 L 列表为由密钥得到的向量 S。根据密钥长度不同，S 盒可由 2、3 或 4bit 的密钥材料得到。对于 128bit 的密钥，相应 S 盒为：

$s_0(x) = q_1[q_0[q_0[x] \oplus s_{0,0}] \oplus s_{1,0}]$

$s_1(x) = q_0[q_0[q_1[x] \oplus s_{0,1}] \oplus s_{1,1}]$

$s_2(x) = q_1[q_1[q_0[x] \oplus s_{0,2}] \oplus s_{1,2}]$

$s_3(x) = q_0[q_1[q_1[x] \oplus s_{0,3}] \oplus s_{1,3}]$

（4）密钥扩展

扩展密钥也是由 h 函数得到的，过程为：

$\rho = 2^{24} + 2^{16} + 2^8 + 2^0$

$A_i = h(2i\rho, M_e)$

$B_i = ROL(h((2i+1)\rho, M_o), 8)$

$K_{2i} = (A_i + B_i) \bmod 2^{32}$

$K_{2i+1} = ROL((A_i + 2B_i) \bmod 2^{32}, 9)$

综上所述，每一轮中的 F 函数可表示为如图 3-9 所示。

3.4.2 Twofish 安全性分析

Bruce Schneier 等人采用多种攻击手段对 Twofish 的安全性进行了测试，包括差分攻击、线性攻击、非线性分析、差分-线性分析、篡改攻击、部分密钥猜测攻击、相关密钥分析、旁路攻击及误差分析等，得到的结论是完整的 20 轮 Twofish 算法能够有效地抵制上述攻击。其中，使用 $2^{22.5}$ 个选择（明文，密文）对，进行 2^{51} 次 g 函数运算，只能成功破译 5 轮不进行输出白化操作的 Twofish 运算；而对 10 轮没有输入白化和输出白化操作的 Twofish 算法进行选择密钥攻击需要 2^{32} 个选择（明文，密文）对或 2^{11} 个自适应选择（明文，密文）对，

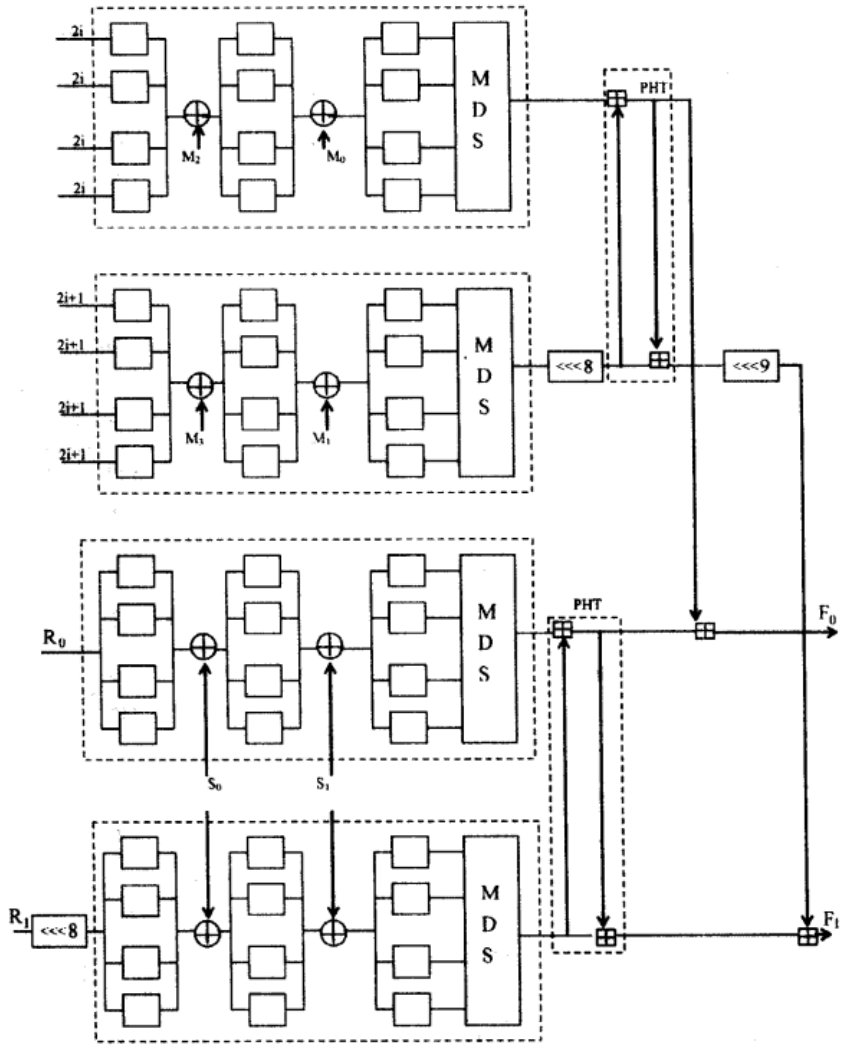


图 3-9　F 函数

工作载荷约为 2^{32}；对 12 轮 Twofish 算法进行差分攻击，成功的概率约为 $2^{-102.8}$。

3.4.3　Twofish 性能测试

（1）在大型微处理器上的实现性能

表 3-9 列出了该算法在不同的密钥长度，不同的运行环境下，采用不同的语言实现时的性能测试结果。

表 3-9　具有不同密钥长度的 Twofish 算法性能

处理器	语言	密钥方案	代码大小	密钥建立时钟数			加密时钟数		
				128bit	192bit	256bit	128bit	192bit	256bit
PentiumPro/Ⅱ	Assembly	编译	8900	12700	15400	18100	285	285	285
PentiumPro/Ⅱ	Assembly	完全	8450	7800	10700	13500	315	315	315

（续）

处理器	语言	密钥方案	代码大小	密钥建立时钟数			加密时钟数		
				128bit	192bit	256bit	128bit	192bit	256bit
PentiumPro/Ⅱ	Assembly	部分	10700	4900	7600	10500	460	460	460
PentiumPro/Ⅱ	Assembly	最小	13600	2400	5300	8200	720	720	720
PentiumPro/Ⅱ	Assembly	零	9100	1250	1600	2000	860	1130	1420
PentiumPro/Ⅱ	MS C	完全	11200	8000	11200	15700	600	600	600
PentiumPro/Ⅱ	MS C	部分	13200	7100	9700	14100	800	800	800
PentiumPro/Ⅱ	MS C	最小	16600	3000	7800	12200	1130	1130	1130
PentiumPro/Ⅱ	MS C	零	10500	2450	3200	4000	1310	1750	2200
PentiumPro/Ⅱ	Borland C	完全	14100	10300	13600	18800	640	640	640
PentiumPro/Ⅱ	Borland C	部分	14300	9500	11200	16600	840	840	840
PentiumPro/Ⅱ	Borland C	最小	17300	4600	10300	15300	1160	1160	1160
PentiumPro/Ⅱ	Borland C	零	10100	3200	4200	4800	1910	2670	3470
Pentium	Assembly	编译	8900	24600	26800	28800	290	290	290
Pentium	Assembly	完全	8200	11300	14100	16000	315	315	315
Pentium	Assembly	部分	10300	5500	7800	9800	430	430	430
Pentium	Assembly	最小	12600	3700	5900	7900	740	740	740
Pentium	Assembly	零	8700	1800	2100	2600	1000	1300	1600
Pentium	MS C	完全	11800	11900	15100	21500	630	630	630
Pentium	MS C	部分	14100	9200	13400	19800	900	900	900
Pentium	MS C	最小	17800	3800	11100	16900	1460	1460	1460
Pentium	MS C	零	11300	2800	3900	4900	1740	2260	2760
Pentium	Borland C	完全	12700	14200	18100	26100	870	870	870
Pentium	Borland C	部分	14200	11200	16500	24100	1100	1100	1100
Pentium	Borland C	最小	17500	4700	12100	19200	1860	1860	1860
Pentium	Borland C	零	11800	3700	4900	6100	2150	2730	3270
UltraSPARC	C	完全		16600	21600	24900	750	750	750
UltraSPARC	C	部分		8300	13300	19900	930	930	930
UltraSPARC	C	最小		3300	11600	16600	1200	1200	1200
UltraSPARC	C	零		1700	3300	5000	1450	1680	1870
PowerPC750	C	完全		12200	17100	22200	590	590	590
PowerPC750	C	部分		7800	12200	17300	780	780	780
PowerPC750	C	最小		2900	9100	14200	1280	1280	1280
PowerPC750	C	零		2500	3600	4900	1030	1580	2040
68040	C	完全	16700	53000	63500	96700	3500	3500	3500
68040	C	部分	18100	36700	47500	78500	4900	4900	4900
68040	C	最小	23300	11000	40000	71800	8150	8150	8150
68040	C	零	16200	9800	13300	17000	6800	8600	10400

表 3-9 中的密钥方案分别说明如下：

完全：该选项使用 4Kbit 的表空间进行完全的密钥预计算，每一个 S 盒均扩展为 8 × 32bit 的表，既要进行 S 盒查找，也要与 MDS 矩阵按列相乘，g 函数的计算包括 4 次表查询和 3 次 XOR 操作。在这种情况下，加密和解密的速度是固定的，与密钥大小无关。

部分：若一个密钥仅加密为数不多的几个报文分组，就没有必要进行一次完整的密钥建立过程。该选项预先计算出 4 个 S 盒，分别为 8 × 8bit 的表，MDS 乘法采用 4 个固定的 8 × 32bit 的 MDS 表。这可以将表空间降至 1Kbit。对字节来说，最后一次 q 盒查询已包含在 MDS 表中，在密钥建立过程中只有 k 个 q 盒参与计算 8 × 8bit S 盒。

最小：若一个密钥仅加密极少数的报文分组，可以采用最小密钥建立方案，与部分密钥相比，在计算 S 盒过程中会少计算一层 q 盒，它们在加密过程中计算得到。

零：该方案不需预计算任何 S 盒，因此不需要额外的表空间，所需的每一项都是实时计算的，报文加密时间是密钥建立时间与采用零密钥方案加密时间的总和。

编译：该选项仅适用于采用汇编语言实现方案。子密钥计算时所需的常数直接写在代码中，这样即节省了内存存取时间，也使得利用 Pentium 可以在一个时钟周期内同时完成 PHT 和子密钥加运算。

（2）智能卡实现性能

6805 处理器是一个典型的智能卡处理器，Twofish 在 6805 处理器上的实现情况如表 3-10 所示。

表 3-10　Twofish 在智能卡上的实现

内存/B	代码及表空间	时钟周期/明密文块	处理时间（ms）/明密文块（4MHz）
60	2200	26500	6.6
60	2150	32900	8.2
60	2000	35000	8.7
60	1760	37100	9.3

（3）硬件实现性能

在 Twofish 的硬件实现上，可以有多种可能的空间-时间折中方案。如，可以预先计算每一轮的子密钥，并将它们存在 RAM 上，也可以实时计算这些子密钥。如果实时计算，h 函数的实现逻辑所用时间将加倍。该方案以速度为代价节省了空间，当然也可以增加逻辑门，使实现速度提高两倍。如果预先计算子密钥，在密钥建立阶段必须运行 h 函数以计算子密钥。该方案节省了门电路但却增加了启动时间。再比如，可以预先计算 S 盒，并将它们存储在 RAM 上，这可以提高计算速度，但增加 RAM 意味着逻辑实现单元将增加 1 ~ 2 倍，同时密钥变化时要花费大量的时间去初始化该 RAM。

表 3-11 给出了采用 128bit 密钥时的硬件实现空间及速度。

表 3-11　采用 128bit 密钥时的硬件实现空间及速度

门电路数量	h 块	时钟周期/块	流水线级数	处理速度/MHz	带宽/（Mbit/s）	启动周期
14000	1	64	1	40	80	4
19000	1	32	1	40	160	40

（续）

门电路数量	h 块	时钟周期/块	流水线级数	处理速度/MHz	带宽/(Mbit/s)	启动周期
23000	2	16	1	40	320	20
26000	2	32	2	80	640	20
28000	2	48	3	120	960	20
30000	2	64	4	150	1200	20
80000	2	16	1	80	640	300

在 Twofish 技术报告 3 中，Doug Whiting、Bruce Schneier 给出了更多的性能测试数据，在此略。

3.5 Serpent 密码算法

作为 AES 提交算法之一，Serpent 的前身是 Serpent-0 算法，该算法发表于第五届快速软件加密国际大会（5th International Workshop on Fast Software Encryption），它采用 DES 算法的 S 盒，若使用 192 或 256bit 密钥，其运算速度和 DES 一样快，但安全性却比三重 DES 还要强。Serpent 是 Serpent-0 的改进方案，主要是采用了强度更大的 S 盒，并对子密钥方案作了少许改进。

3.5.1 Serpent 算法描述

Serpent 算法采用 SP 网络，分组长度为 128bit。该算法将明文分为 4 个 32bit 字进行 32 轮运算，在 33 个 128bit 子密钥的控制下得到 128bit 的密文（表示方式为低位在前）。原则上，密钥长度可为任意长，但为满足 AES 的要求，该算法将密钥长度定为 128、192 或 256bit，不足 256bit 的短密钥通过在其末尾附加“1”，其后跟随足够多的“0”，来凑成 256bit。该算法由初始置换 IP、32 轮迭代运算以及末置换 FP 构成。设明文 P 经初始置换 IP 后输出为 B_0，将其作为第一轮的输入（轮数以 0～31 表示），第 i 轮的输出为 B_{i+1}，每一轮的迭代过程用轮函数 R_i 表示，最后一轮的输出为 B_{32}，对其进行末置换得到密文 C。该过程可表示为：

$B_0 = IP\ (P)$

$B_{i+1} = R_i\ (B_i)$

$C = FP\ (B_{32})$

（1）IP 和 FP

同 DES 一样，初始置换 IP 和末置换 FP 并没有密码学上的意义，其线性变换表分别如表 3-12 和表 3-13 所示。

表 3-12 初始置换 IP

0	32	64	96	1	33	65	97	2	34	66	98	3	35	67	99
4	36	68	100	5	37	69	101	6	38	70	102	7	39	71	103

（续）

8	40	72	104	9	41	73	105	10	42	74	106	11	43	75	107
12	44	76	108	13	45	77	109	14	46	78	110	15	47	79	111
16	48	80	112	17	49	81	113	18	50	82	114	19	51	83	115
20	52	84	116	21	53	85	117	22	54	86	118	23	55	87	119
24	56	88	120	25	57	89	121	26	58	90	122	27	59	91	123
28	60	92	124	29	61	93	125	30	62	94	126	31	63	95	127

表 3-13　末置换 FP

0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
64	68	72	76	80	84	88	92	96	100	104	108	112	116	120	124
1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61
65	69	73	77	81	85	89	93	97	101	105	109	113	117	121	125
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62
66	70	74	78	82	86	90	94	98	102	106	110	114	118	122	126
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63
67	71	75	79	83	87	91	95	99	103	107	111	115	119	123	127

（2）迭代运算

Serpent 运算共有 32 轮迭代，每一轮迭代都包含密钥混合操作、S 盒运算、线性置换（最后一轮除外）及密钥加混合运算（最后一轮），即：

$R_i(X) = L(S_i(X \oplus K_i))$，$i = 0, \cdots, 30$

$R_i(X) = S_i(X \oplus K_i) \oplus K_{32}$，$i = 31$

其中，S 盒是一个 4bit 的置换，必须满足如下差分、线性性质：

1）每一个差分特征的最大概率为 1/4，一个单比特的输入差别永远不会造成一个单比特的输出差别。

2）每一个线性特征的概率在（1/2 - 1/4）~（1/2 + 1/4）之间，在一个单比特输入和一个单比特输出之间的线性相关性的概率在（1/2 - 1/8）~（1/2 + 1/8）之间。

3）作为输入位的函数，输出位的排列达到最大值，即 3。

在每一轮中都有 32 个相同的 S 盒并行运行，以第一轮迭代 R_0 为例，$B_0 \oplus K_0$ 的前 4 位 0、1、2、3 作为第一个 S_0 盒的输入，置换输出结果作为中间变量的前 4 位；而第二个 S_0 盒以 $B_0 \oplus K_0$ 的第 4、5、6、7 位作为输入，输出作为中间变量的后 4 位，依此类推。之后对该 128bit 的中间变量进行线性变换。同样，第二轮迭代 R_1 利用 32 个 S1 盒并行地对 $B_1 \oplus K_1$ 进行置换，最后再作线性变换，得到 B_2。

经 S 盒变换后得到的 4 个 32bit 字要进行线性混合，其线性变换过程为：

$X_0, X_1, X_2, X_3 := S_i(B_i \oplus K_i)$

$X_0 := X_0 \lll 13$

$X_2 := X_2 \lll 3$

$X_1 := X_1 \oplus X_0 \oplus X_2$

$X_3 := X_3 \oplus X_2 \oplus (X_0 \ll 3)$

$X_1 := X_1 \lll 1$

$X_3 := X_3 \lll 7$

$X_0 := X_0 \oplus X_1 \oplus X_3$

$X_2 := X_2 \oplus X_3 \oplus (X_1 \ll 7)$

$X_0 := X_0 \lll 5$

$X_2 := X_2 \lll 22$

$B_{i+1} := X_0, X_1, X_2, X_3$

若以比特流的形式表示，则线性变换过程中各位的变化如表 3-14 所示。

表 3-14　比特流的线性变换

{16 52 56 70 83 94 105}	{72 114 125}	{2 9 15 30 76 84 126}	{ 36 90 103}
{20 56 60 74 87 98 109}	{1 76 118}	{2 6 13 19 34 80 88}	{40 94 107}
{24 60 64 78 91 102 113}	{5 80 122}	{6 10 17 23 38 84 92}	{44 98 111}
{28 64 68 82 95 106 117}	{9 84 126}	{10 14 21 27 42 88 96}	{48 102 115}
{32 68 73 86 99 110 121}	{2 13 88}	{14 18 25 31 46 92 100}	{52 106 119}
{36 72 76 90 103 114 125}	{6 17 92}	{18 22 29 35 50 96 104}	{56 110 123}
{1 40 76 80 94 107 118}	{10 21 96}	{22 26 33 39 54 100 108}	{60 114 127}
{5 44 80 84 98 111 122}	{14 25 100}	{26 30 37 43 58 104 112}	{3 118}
{9 48 84 88 102 115 126}	{18 29 104}	{30 34 41 47 62 108 116}	{7 122}
{2 13 52 88 92 106 119}	{22 33 108}	{34 38 45 51 66 112 120}	{11 126}
{6 17 56 92 96 110 123}	{26 37 112}	{38 42 49 55 70 116 124}	{2 15 76}
{10 21 60 96 100 114 127}	{30 41 116}	{0 42 46 53 59 74 120}	{6 19 80}
{3 14 25 100 104 108}	{34 45 120}	{4 46 50 57 63 78 124}	{10 23 84}
{7 18 29 104 108 122}	{38 49 124}	{0 8 50 54 61 67 82}	{14 27 88}
{11 22 33 108 112 126}	{0 42 53}	{4 12 54 58 65 71 86}	{18 31 92}
{2 15 26 37 76 112 116}	{4 46 57}	{8 16 58 62 69 75 90}	{22 35 96}
{6 19 30 41 80 116 120}	{8 50 61}	{12 20 62 66 73 79 94}	{26 39 100}
{10 23 34 45 84 120 124}	{12 54 65}	{12 24 66 70 77 83 98}	{30 43 104}
{0 14 27 38 49 68 124}	{16 58 69}	{20 28 70 74 81 87 102}	{34 47 108}
{0 4 18 31 42 53 92}	{20 62 73}	{24 32 74 78 85 91 106}	{38 51 112}
{4 8 22 35 46 57 96}	{24 66 77}	{28 36 78 82 89 95 110}	{42 55 116}
{8 12 26 39 50 61 100}	{28 70 81}	{32 40 82 86 93 99 114}	{46 59 120}
{12 16 30 43 54 65 104}	{32 74 85}	{36 90 103 118}	{50 63 124}
{16 20 34 47 58 69 108}	{36 78 89}	{40 94 107 122}	{0 54 67}
{20 24 38 51 62 73 112}	{40 82 93}	{44 98 111 126}	{4 58 71}
{24 28 42 55 66 77 116}	{44 86 97}	{2 48 102 115}	{8 62 75}
{28 32 46 59 70 81 120}	{48 90 101}	{6 52 106 119}	{12 66 79}
{32 36 50 63 74 85 124}	{52 94 105}	{10 56 110 123}	{16 70 83}
{0 36 40 54 67 78 89}	{56 98 109}	{14 60 114 127}	{20 74 87}
{4 40 44 58 71 82 93}	{60 102 113}	{3 18 72 114 118 125}	{24 78 91}
{8 44 48 62 75 86 97}	{64 106 117}	{1 7 22 76 118 122}	{28 82 95}
{12 48 52 66 79 90 101}	{68 110 121}	{5 11 26 80 122 126}	{32 86 99}

在上表中，输出比特为相应位置中括号内的输入比特的异或，比如输出比特 1 为输入的第 72、114 和 125 比特的异或。

在 Serpent 中共有 8 个不同的 S 盒 S_0，…，S_7，因此在 32 轮迭代中它们将被使用 4 次，

这样在第 7 轮使用 S_7 后，第 8 轮将再次使用 S_0，第 9 轮使用 S_1，……

Serpent 的 S 盒生成利用了一个 32 × 16 的矩阵，该矩阵初始值为 32 行 DES 的 S 盒，根据第（$r+1$）行的值和密钥的值来置换第 r 行的值，如果得到的行满足 Serpent S 盒所要求的差分和线性性质，就将其作为一个 S 盒，重复上述过程，直到产生所需的 S 盒。比如，对于一个 16B 的分组，将其前 4bit 分别作为数组 Serpent［16］的元素，用 sbox［32］［16］存储 8 个 DES 的 S 盒，swapentries（，）表示行置换，则生成 S 盒的伪代码为：

```
index: =0;
repeat
    currentsbox: = index modulo 32;
    for i: =0 to 15 do
        j: =sbox [ (currentsbox +1) modulo 32] [serpent [i]];
        swapentries (sbox [currentsbox] [i], sbox [currentsbox] [j]);
    if sbox [currentsbox] [ . ] satisfies the desired properties, save it;
    index: = index +1;
until 8 S-boxes have been generated.
```

8 个 S 盒如表 3-15 所示。

表 3-15　Serpent S 盒

S0	3	8	15	1	10	6	5	11	14	13	4	2	7	0	9	12
S1	15	12	2	7	9	0	5	10	1	11	14	8	6	13	3	4
S2	8	6	7	9	3	12	10	15	13	1	14	4	0	11	5	2
S3	0	15	11	8	12	9	6	3	13	1	2	4	10	7	5	14
S4	1	15	8	3	12	0	11	6	2	5	4	10	9	14	7	13
S5	15	5	2	11	4	10	9	12	0	3	14	8	13	6	7	1
S6	7	2	12	5	8	4	6	11	14	9	1	15	13	3	10	0
S7	1	13	15	0	14	8	2	11	7	4	12	10	9	3	5	6

（3）子密钥生成方案

Serpent 的初始密钥经填充后为 256bit，采用密钥扩展方案，要将其扩展为 33 个 128bit 的子密钥 K_0，…，K_{32}。将初始密钥写成 8 个 32bit 的字 w_{-8}，…，w_{-1}，中间子密钥字 w_0，…，w_{131}的生成为：

$$w_i := (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \phi \oplus i) \lll 11$$

其中，ϕ 为黄金分割率（$\sqrt{5}+1$）/2 的小数部分，用 16 进制表示为 0x9e3779b9。

各轮子密钥的生成也要用到 S 盒，将子密钥字 w_0，…，w_{131}变换为子密钥，方式如下：

$$\{k_0, k_1, k_2, k_3\} := S_3(w_0, w_1, w_2, w_3)$$

$$\{k_4, k_5, k_6, k_7\} := S_2(w_4, w_5, w_6, w_7)$$

$$\{k_8, k_9, k_{10}, k_{11}\} := S_1(w_8, w_9, w_{10}, w_{11})$$

$$\{k_{12}, k_{13}, k_{14}, k_{15}\} := S_0(w_{12}, w_{13}, w_{14}, w_{15})$$

$$\{k_{16}, k_{17}, k_{18}, k_{19}\} := S_7(w_{16}, w_{17}, w_{18}, w_{19})$$

$$\vdots$$

$\{k_{124}, k_{125}, k_{126}, k_{127}\} := S_4(w_{124}, w_{125}, w_{126}, w_{127})$

$\{k_{128}, k_{129}, k_{130}, k_{131}\} := S_3(w_{128}, w_{129}, w_{130}, w_{131})$

$K_i' = \{k_{4i}, k_{4i+1}, k_{4i+2}, k_{4i+3}\}$

最后，对 K_i'进行初始置换，得到各轮子密钥为：$K_i = IP(K_i')$。

（4）解密

解密是加密过程的逆过程，主要表现在解密时要逆序使用各 S 盒，且解密 S 盒是加密 S 盒的逆，同时解密的线性变换是加密变换的逆函数，解密子密钥的使用顺序是加密子密钥的逆序。逆线性变换及 S 盒的逆分别如表 3-16 及表 3-17 所示。

表 3-16　逆线性变换

{53 55 72}	{1 5 20 90 }	{ 15 102}	{3 31 90}
{57 59 76}	{5 9 24 94 }	{ 19 106}	{7 35 94}
{61 63 80}	{9 13 28 98 }	{ 23 110}	{11 39 99}
{65 67 84}	{13 17 32 102 }	{ 27 114}	{1 3 15 20 43 102}
{69 71 88}	{17 21 36 106 }	{1 31 118}	{5 7 19 24 47 106}
{73 75 92}	{21 25 40 110}	{5 35 122}	{9 11 23 28 51 110}
{77 79 96}	{25 29 44 114}	{9 39 126}	{13 15 27 32 55 114}
{81 83 100}	{1 29 33 49 118}	{2 13 43}	{1 17 19 31 36 59 118}
{85 87 104}	{5 33 37 52 122}	{6 17 47}	{5 21 23 35 40 63 122}
{89 91 108}	{9 37 41 56 126}	{10 21 51}	{9 25 27 39 44 67 126}
{93 95 112}	{2 13 41 45 60}	{14 25 55}	{2 13 29 31 43 48 71}
{97 99 116}	{6 17 45 49 64}	{18 29 59}	{6 17 33 35 47 52 75}
{101 103 120}	{10 21 49 53 68}	{22 33 63}	{10 21 37 39 51 56 79}
{105 107 124}	{14 25 53 57 72}	{26 37 67}	{14 25 41 43 55 60 83}
{0 109 111}	{18 29 57 61 76}	{30 41 71}	{18 29 45 47 59 64 87}
{4 113 115}	{22 33 61 65 80}	{34 45 75}	{22 33 49 51 63 68 91}
{8 117 119}	{26 37 65 69 84}	{38 49 79}	{26 37 53 55 67 72 95}
{12 121 123}	{30 41 69 73 88}	{42 53 83}	{30 41 57 59 71 76 99}
{16 125 127}	{34 45 73 77 92}	{46 57 87}	{34 45 61 63 75 80 103}
{1 3 20}	{38 49 77 81 96}	{50 61 91}	{38 49 65 67 79 84 107}
{5 7 24}	{42 53 81 85 100}	{54 65 95}	{42 53 69 71 83 88 111}
{9 11 28}	{46 57 85 89 104}	{58 69 99}	{46 57 73 75 87 92 115}
{13 15 32}	{50 61 89 93 108}	{62 73 103}	{50 61 77 79 91 96 119}
{17 19 36}	{54 65 93 97 112}	{66 77 107}	{54 65 81 83 95 100 123}
{21 23 40}	{58 69 97 101 116}	{70 81 111}	{58 69 85 87 99 104 127}
{25 27 44}	{62 73 111 105 120}	{74 85 115}	{3 62 73 89 91 103 108}
{29 31 48}	{66 77 115 109 124}	{78 89 119}	{7 66 77 93 95 107 112}
{33 35 52}	{0 70 81 109 113}	{82 93 113}	{11 70 81 97 99 111 116}
{37 39 56}	{4 74 85 113 117}	{86 97 127}	{15 74 85 101 103 115 120}
{41 43 60 }	{8 78 89 117 121}	{ 3 90}	{19 78 89 105 107 119 124}
{45 47 64}	{12 82 93 121 125}	{ 7 94}	{0 23 82 93 109 111 123}
{49 51 68}	{1 16 86 97 125}	{ 11 98}	{4 27 86 97 113 115 127}

表 3-17　S 盒的逆

InvS0	13	3	11	0	10	6	5	12	1	14	4	7	15	9	8	2
InvS1	5	8	2	14	15	6	12	3	11	4	7	9	1	13	10	0
InvS2	12	9	15	4	11	14	1	2	0	3	6	13	5	8	10	7
InvS3	0	9	10	7	11	14	6	13	3	5	12	2	4	8	15	1
InvS4	5	0	8	3	10	9	7	14	2	12	11	6	4	15	13	1
InvS5	8	15	2	9	4	1	13	14	11	6	5	3	7	12	10	0
InvS6	15	10	1	13	5	3	6	0	4	9	14	7	2	12	8	11
InvS7	3	0	6	13	9	14	15	8	5	12	11	7	10	1	4	2

3.5.2　Serpent 安全性分析

Ross Anderson 等人在提交 Serpent 算法时也对该算法的安全性进行了分析，并分别就采用字典攻击、密钥碰撞攻击、差分分析、线性分析、相关密钥分析、时间攻击、电磁泄漏攻击、误差分析等攻击方式时的算法强度进行了说明。他们预计不论采用何种攻击手段所需的已知（明文，密文）对或选择（明文，密文）对均超过 2^{100}。

采用穷举攻击时，该算法的强度如表 3-18 所示：

表 3-18　采用穷举攻击时算法的强度

分组大小/bit	密钥长度/bit	搜索空间/bit	选择/已知明密文对
128	128	2^{128}	1
128	192	2^{192}	2
128	256	2^{256}	2

Orr Dunkelman 对 Serpent 算法的变形 Serpent-p、Serpent-p-ns 进行了差分分析及线性分析，并给出了详细的说明，从而也证明了 Serpent 算法是安全的。Tadayoshi Kohno 等人采用差分攻击、Boomerang 攻击、加强 Boomerang 攻击及中间截击攻击等迄今对 Serpent 算法最好的攻击手段也只能对轮数较少的 Serpent 算法进行成功破译，并且是以极大的数据复杂性、工作负载及内存消耗为代价的。详细结果可参考有关文献，在此不作详述。

3.5.3　Serpent 性能测试

在 133MHz 的 Pentium/MMX 处理器上，Serpent 每秒加密 9791000bit，与 DES 速度相仿（在同样的机器上，采用同样的测试程序，DES 在最优化实现情况下，每秒加密 9824864bit）。在其他 32bit 处理器上，Serpent 一般使用 1830～1940 条指令加密 128bit，而标准 DES 加密 64bit 也需 685 条指令。

在硬件实现上，若采用流机制至多使用 100000 个逻辑门，其中加密、解密、子密钥计算各需 33000 个门，还包括一些控制逻辑和缓存。

3.6　Rijndael 密码算法

Rijndael 算法设计时主要考虑到要能够抵抗各种已知的攻击方式、在多种平台上实现时

的速度以及设计的简单性。与其他分组算法不同，Rijndael 算法的原形是 Square 算法，设计策略是基于宽轨迹策略的。宽轨迹策略能够有效地抵制线性分析和差分分析。该过程分 3 层：线性混合层保证经过多轮运算后具有高的扩散度；非线性层采用并行 S 盒运算，确保非线性性质，起到混淆作用；密钥层运算将各轮子密钥与中间状态异或。

3.6.1 Rijndael 算法描述

Rijndael 算法的分组长度和密钥长度都是可变的，可取 128、192 或 256bit。在 Rijndael 中，将运算的中间结果称为状态。状态可用矩阵表示，该矩阵有 4 行，矩阵的列用 *Nb* 表示，*Nb* 等于分组长度除以 32 的商，如图 3-10 所示。

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

图 3-10　Nb = 6 时的状态

同样，密钥也可用字节矩阵的形式表示，矩阵的列数 Nk 为密钥长度除以 32 的商，如图 3-11。

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

图 3-11　Nk = 4 时的密钥

有时矩阵也可用一维向量数组表示，每个向量元素为 4B，由矩阵的列构成，实际上是构成了一个字。在加密过程中，明文对应着状态矩阵的 $a_{0,0}$、$a_{1,0}$、$a_{2,0}$、$a_{3,0}$、$a_{0,1}$、$a_{1,1}$、$a_{2,1}$、$a_{3,1}$，…，加密密钥对应着密钥矩阵 $k_{0,0}$、$k_{1,0}$、$k_{2,0}$、$k_{3,0}$、$k_{0,1}$、$k_{1,1}$、$k_{2,1}$、$k_{3,1}$、…，加密结束后将状态矩阵中的数据按同样的顺序排列构成密文输出。因此，若在分组块中一个字节的序号为 n，该字节在矩阵中的对应位置为（i，j），则

$$i = n \bmod 4, j = [n/4], n = i + 4j$$

其中，i 为矩阵列号，j 为该字节矩阵中的行号。

Rijndael 算法的迭代轮数 Nr 并不是固定的，根据 Nb 和 Nk 的不同而不同，如表 3-19 所示。

表 3-19　迭代轮数 Nr

Nr	Nb = 4	Nb = 6	Nb = 8
Nk = 4	10	12	14
Nk = 6	12	12	14
Nk = 8	14	14	14

Rijndael 算法包括 3 步：初始密钥加、Nr - 1 步轮迭代以及最后一轮线性混合。在进行迭代运算之前先要在密钥加层次上进行明文与密钥的异或运算；而最后一轮迭代运算与前面各轮不同，主要是为了保证加密与解密在结构上一致。

（1）Rijndael 轮变换

Rijndael 的迭代运算可用伪代码表示如下：

```
Round(State,RoundKey)
{
  ByteSub(State);
  ShiftRow(State);
  MixColumn(State);
  AddRoundKey(State,RoundKey);
}
```

最后一轮迭代稍有不同，可描述为：

```
FinalRound(State,RoundKey)
{
  ByteSub(State);
  ShiftRow(State);
  AddRoundKey(State,RoundKey);
}
```

可将轮迭代表示为图 3-12。

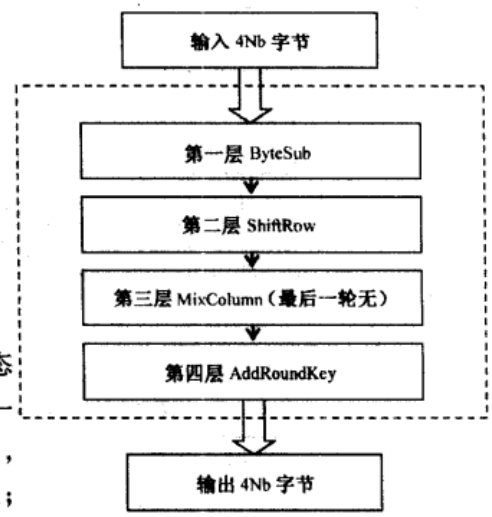


图 3-12　轮迭代

1）第一层：ByteSub。

该层为一个非线性字节替换，独立地在每个状态字节上进行 S 盒运算。S 盒运算过程为：首先将每一个非 0 字节表示成有限域 GF（2^8）上的元素形式，然后将它们替换为各自的乘法逆元，0 字节保持不变；之后在域 GF（2）上进行仿射变换，将其与一个矩阵相乘，并与（1 1 0 0 0 1 1 0）相加，即

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

S 盒的作用如图 3-13 所示。

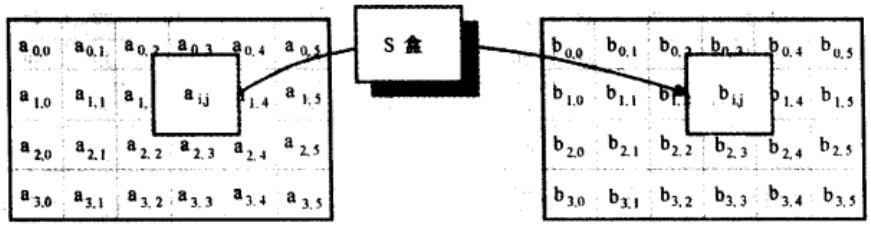


图 3-13　S 盒的替换效果

2）第二层：ShiftRow。

该层主要是将分组中的字节进行置换，通过将状态中的最后 3 行移位不同的位移量得到，各行的位移量与 Nb 有关，如表 3-20 所示。

表 3-20　位移量与 Nb 的关系

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

ShiftRow 对状态的操作效果如图 3-14 所示。

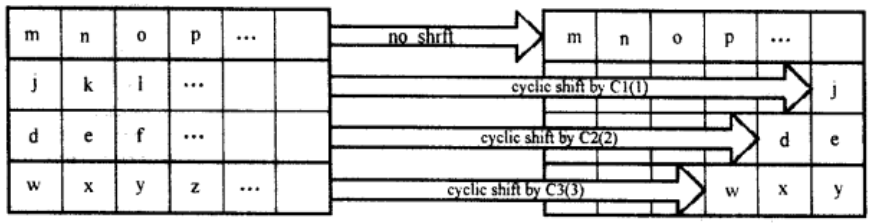


图 3-14　行移位变换对状态中的行操作

ShiftRow 置换的实际效果如表 3-21 ~ 表 3-23 所示。

表 3-21　128bit 分组的 ShiftRow 置换（Nb = 4）

ShiftRow 之前				ShiftRow 之后			
0	4	8	12	0	4	8	12
1	5	9	13	5	9	13	1
2	6	10	14	10	14	2	6
3	7	11	15	15	3	7	11

表 3-22　192bit 分组的 ShiftRow 置换（Nb = 4）

ShiftRow 之前						ShiftRow 之后					
0	4	8	12	16	20	0	4	8	12	16	20
1	5	9	13	17	21	5	9	13	17	21	1
2	6	10	14	18	22	10	14	18	22	2	6
3	7	11	15	19	23	15	19	23	3	7	11

表 3-23　256bit 分组的 ShiftRow 置换（Nb = 4）

ShiftRow 之前								ShiftRow 之后							
0	4	8	12	16	20	24	28	0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29	5	9	13	17	21	25	29	1
2	6	10	14	18	22	26	30	14	18	22	26	30	2	6	10
3	7	11	15	19	23	27	31	19	23	27	31	3	7	11	15

从以上 3 个表可以看到，第一行不进行置换，而对于位于第 i（i = 2，3，4）行的字节元素，其位置将左移 C_{i-1} 位，即第 j 列的元素将被置换至（$j - C_{i-1}$）mod Nb 列（j = 1，2，…）。

该过程的逆过程可描述为：在第 i 行（i = 2，3，4）第 j 列的元素将被移至同一行的（j

$+Nb-C_{i-1}$）mod Nb 列。

3）第三层：MixColumn。

在 MixColumn 层中，状态矩阵的各列可看作是在域 GF（2^8）上的多项式，其状态变换是与一个固定多项式 $c(x)$ 模乘的结果，模为 x^4+1。$c(x)=c_3x^3+c_2x^2+c_1x+c_0$，其中各系数分别为 $c_0(x)=x$，$c_1(x)=1$，$c_2(x)=1$，$c_3(x)=x+1$，也可将其表示为 $c(x)=$'03' x^3+'01' x^2+'01' $x+$'02'。由于 $c(x)$ 与 x^4+1 互素，因此是可逆多项式，状态变换可表示成多项式乘法的形式，即 $b(x)=c(x)\times a(x)$，

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

对状态中的各列均实行模乘操作，可表示为 MixColumn（State），如图 3-15 所示。

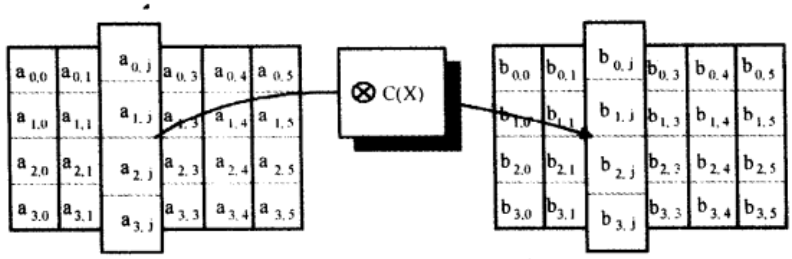


图 3-15　列混淆操作

MixColumn 的逆操作与此类似，也是将状态中的列 $b(x)$ 与一个固定多项式 $d(x)=d_3x^3+d_2x^2+d_1x+d_0$ 模乘，模仍为 x^4+1，$d(x)$ 满足 $c(x)\times d(x)=$'01'，可得 $d_0(x)=x^3+x^2+x$，$d_1(x)=x^3+1$，$d_2(x)=x^3+x^2+1$，$d_3(x)=x^3+x+1$，即 $d(x)=$'0B' x^3+'0D' x^2+'09' $x+$'0E'。相应地，状态变换矩阵为

$$\begin{pmatrix} 0E\ 0B\ 0D\ 09 \\ 09\ 0E\ 0B\ 0D \\ 0D\ 09\ 0E\ 0B \\ 0B\ 0D\ 09\ 0E \end{pmatrix}$$

4）第四层：RoundKeyAddition。

在该阶段的操作中，各轮子密钥与状态矩阵中的各元素进行简单的按位异或运算，子密钥是由加密密钥通过子密钥生成方案得到的，子密钥的长度等于分组长度 Nb。该操作可表示为 AddRoundKey（State，RoundKey），如图 3-16 所示。

图 3-16　轮密钥加操作

该操作的逆操作就是它自身。

(2) 子密钥生成方案

在 Rijndael 中，加/解密均需要 Nr 个子密钥，每个密钥为 4B 的字。可将扩展密钥表示为数组 W [Nb * (Nr + 1)]，前 Nk 的字元素就是加密密钥。密钥扩展方案与 Nk 的值有关，若 Nk≤6，扩展方案可描述为：

```
KeyExpansion(byte Key[4 * Nk], word W[Nb * (Nr + 1)])
{
  for (i = 0;i < Nk;i + + )
    W[i] = (Key[4 * i],Key[4 * i + 1], Key[4 * i + 2], Key[4 * i + 3]);
  for (i = Nk; i < Nb * (Nr + 1);i + + )
  {
    temp = W[i - 1];
    if (i % Nk = =0)
      temp = SubByte (RotByte (temp)) XOR Rcon[i/Nk];
    W[i] = W[i - Nk] XOR temp;
  }
}
```

若 Nk > 6，则有：

```
KeyExpansion(byte Key[4 * Nk], word W[Nb * (Nr + 1)])
{
  for (i = 0;i < Nk;i + + )
    W[i] = (Key[4 * i],Key[4 * i + 1], Key[4 * i + 2], Key[4 * i + 3]);
  for (i = Nk; i < Nb * (Nr + 1);i + + )
  {
    temp = W[i - 1];
    if (i % Nk = =0)
      temp = SubByte (RotByte (temp)) XOR Rcon[i/Nk];
    else if (i % Nk = =4)
      temp = SubByte(temp);
    W[i] = W[i - Nk] XOR temp;
  }
}
```

其中，SubByte (W) 函数返回一个 4B 的字，每个字节是由输入字中的相应字节进行 S 盒变换得到的。RotByte(W)将输入字的字节循环左移 1B，若输入字为(a, b, c, d)，则其输出为(b, c, d, a)。Rcon[i]为每一轮的常数，与 Nk 无关，Rcon[i] = (RC[i],'00','00','00')，RC[i]为 GF(2^8)上的元素 $x^{(i-1)}$，RC[1] = 1 (即'01')，RC[i] = x(即'02') × (RC[i - 1]) = $x^{(i-1)}$。

(3) Rijndael 解密方案

Rijndael 的解密过程与加密过程顺序相反，且在每一子过程中其变换为加密操作的逆。

上节已分别介绍了 ByteSub、ShiftRow 和 MixColumn 的逆运算，在此分别将其表示为 InvByteSub、InvShiftRow 和 InvMixColumn，则解密过程的轮操作为：

```
InvFinalRound(word State, word RoundKey)
{
  AddRoundKey(State,RoundKey);
  InvShiftRow(State);
  InvByteSub(State).
}

InvRound(word State, word RoundKey)
{
  AddRoundKey(State,RoundKey);
  InvMixColumn(State);
  InvShiftRow(State);
  InvByteSub(State).
}
```

Rijndael 的整个解密操作为：

```
InvRijndael (byte State,byte CipherKey)
{
  KeyExpansion (CipherKey, ExpandedKey);
  InvFinalRound (State, ExpandedKey + Nb * Nr);
  for ( i = Nr - 1;i > 0;i - - )
    InvRound (State, ExpandedKey + Nb * i);
  AddRoundKey (State, ExpandedKey);
}
```

3.6.2 Rijndael 安全性分析

通过对 Rijndael 算法及其简化版本进行差分分析、线性分析、Square 攻击及篡改攻击，人们认为对 Rijndael 算法最有效的攻击方式为穷举攻击，当密钥长度分别为 16、24、32B 时，穷举攻击的工作载荷分别为 2^{127}、2^{191}、2^{255}次 Rijndael 运算。

Rijndael 算法的策略是宽轨迹策略，宽轨迹策略是针对差分分析和线性分析提出的，其最大优点是可以给出算法的最佳差分特征的概率以及最佳线性逼近的偏差的界，这对于分析算法抵抗差分密码分析及线性密码分析的能力都是很有帮助的。分析显示，4 轮 Rijndael 的最佳差分特征概率及最佳线性逼近的偏差分别为 2^{-150}和 2^{-76}，8 轮 Rijndael 的最佳差分特征概率及最佳线性逼近的偏差分别为 2^{-300}和 2^{-151}。

“Square”攻击是针对 Square 算法提出的一种攻击方法，该攻击同样适用于 Rijndael。结果表明，7 轮以上的 Rijndael 算法对“Square”攻击是免疫的。

该算法不存在弱密钥或半弱密钥。详细分析请参考相关文献。

3.6.3 Rijndael 性能测试

Rijndael 算法能够在多种处理器上高效实现，也适合于硬件实现。例如在 8 位处理器上，Rijndael 的各变换部件都可以通过简单的编程实现，其中 RowShift 和 AddRoundKey 可直接实现，ByteSub 操作需一个 256B 的表。对于一个状态字节，可有效地连续进行 AddRoundKey、ByteSub 及 RowShift 操作。同样，MixColumn 变换，即在域 GF（2^8）上的矩阵乘法也可通过异或运算及移位操作快速实现。以一列状态为例，运算过程为：

```
Tmp = a[0] XOR a[1] XOR a[2] XOR a[3];
Tm = a[0] XOR a[1]; Tm = xtime (Tm); a[0] = a[0] XOR Tm XOR Tmp;
Tm = a[1] XOR a[2]; Tm = xtime (Tm); a[1] = a[1] XOR Tm XOR Tmp;
Tm = a[2] XOR a[3]; Tm = xtime (Tm); a[2] = a[2] XOR Tm XOR Tmp;
Tm = a[3] XOR a[0]; Tm = xtime (Tm); a[3] = a[3] XOR Tm XOR Tmp;
```

其中，xtime(Word) = Word × '02'（即 x）XOR '1B'。

在 32 位(或以上)处理器中，Rijindael 算法的处理速度更快，因为轮变换的不同步骤都可在一个查询表中统一进行，具体过程可参见相关文献。同时，在每轮变换中也可采用适当的并行处理技术，如在状态字节、行、列等各部分的操作可并行处理。

Rijndael 算法也适合在硬件上实现。

3.7 小结

本章介绍了 AES 的 5 种候选算法，其中 Rijndael 算法最终获胜，成为新的数据加密标准，这主要是因为该算法具有很好的安全性，同时具有性能好、效率高、易用、灵活等优点。该算法采用非线性结构的 S 盒，安全性高；硬、软件实现均表现出较好的性能；密钥安装时间短，灵活性高；内存需求低，适合于受限环境；操作简单，并可抵御能量攻击、时间攻击等多种密码分析方式；另外，算法的分组长度及密钥长度的设计很灵活，根据不同的分组长度和密钥长度，算法的迭代次数也是不同的。

3.8 习题

1. 在设计 AES 算法时的基本要求是什么？为什么 Rijndael 算法被选定为 AES？

2. 按照 FIPS—197 规范，熟悉 AES 主要算法流程，并编程实现单分组加/解密功能模块。

3. 在习题 2 实现的功能模块基础上，编程实现 AES 在 4 种操作模式下的加/解密方案。

4. Rijndael 算法的策略是宽轨迹策略，请查找相关文献资料，了解宽轨迹策略的设计思想，并对该策略的安全性进行分析。

5. 查阅资料，了解当前针对新一轮 AES 算法标准的主要攻击手段及攻击成果，结合相关防御措施，形成综述报告。

6. 结合 Twofish 算法流程，解释如何基于 Feistel 网络设计分组密码算法。

7. 结合 Rijndael 及 Serpent 算法流程，解释基于 SP 网络设计分组密码算法的主要过程。

8. 结合 Mars 及 RC6 算法流程，了解使用扩展 Feistel 网络设计分组密码算法的主要过程。

第 4 章　椭圆曲线密码

1976 年，Diffie 和 Hellman 提出了公钥密码学（Public-key Cryptography），离散对数问题（DLP）和密码学意义上的重要性开始逐渐为人们所重视起来。ElGamal 首先描述出如何将离散对数问题应用到公钥加密和数字签名方案中去。随后，ElGamal 的这种方法被进一步改善提高，并且被各种各样的应用中的各种各样的协议所采纳。其中之一就是美国政府的数字签名算法（DSA）。

尽管当年 Diffie 和 Hellman 提出离散对数问题是要应用到他们的密钥协商协议中，这时离散对数问题被精确定义为寻找模 p 乘法群中生成元的对数问题，但是现在这个问题可以被推广到任意群中。椭圆曲线密码系统正是基于有限域上的椭圆曲线点所构成的群的离散对数问题。

椭圆曲线已经被广泛研究了 100 多年，而且从中得到了非常广泛的研究课题。椭圆曲线现在已经成为很多重要应用领域的工具，如编码理论、伪随机比特生成以及数论算法（素性证明、整数分解）等。

椭圆曲线密码系统自 1985 年由 Neal Koblitz 和 Victor Miller 分别独立地提出，其后人们对它的安全性和实现的有效性进行了广泛和深入地研究。椭圆曲线密码系统是建立在椭圆曲线点群的离散对数问题上的。在有限域上椭圆曲线点群中，还没有关于寻找离散对数的诸如“Index-calculus”的亚指数时间算法出现。因此，利用规模更小的椭圆曲线群来达到相同的安全级别，可以拥有更小的密钥长度、更小的带宽需要，以及更快的实现。这些特性对于那些计算能力和集成芯片空间受限的安全应用特别具有吸引力，例如，在智能卡、PC（Personal Computer）卡，以及无线设备上。

4.1　基本概念

4.1.1　群

群是一种拥有运算的集合 G，该运算 $\circ$ 满足如下性质：

1）封闭：对于 G 中任意的 x、y，$x \circ y$ 属于 G。

2）结合律：对于所有的 x、y、z，有 $(x \circ y) \circ z = x \circ (y \circ z)$。

3）单位元：G 存在这样的 e，使得对于 G 中的所有元素 x，都有 $x \circ e = e \circ x = x$。

4）逆元：对于 G 中任意的元素 x，存在 G 中元素 y，使得 $x \circ y = y \circ x = e$。

从上面可以得到交换的性质。

5）交换性：对于 G 所有的元素 x、y，有 $x \circ y = y \circ x$，这时，称这个群是一个阿贝尔群，也就是交换群。

通常，运算 $x \circ y$ 可简写成 xy。

【例 4-1】 整数集合 **Z** 对于通常的加法构成一个交换群，但对通常的乘法不构成一个

群。

【例 4-2】 有理数集合 **Q** 对于通常的加法构成一个交换群，对通常的乘法也构成一个交换群。

【例 4-3】 实数集合 **R** 对于通常的加法构成一个交换群，对通常的乘法也构成一个交换群。

【例 4-4】 设 n 是正整数，对任意整数 a，a 被 n 除的最小非负余数为 $a \bmod n$。由全体元素 $a \bmod n$ 组成的集合记为 $\mathbf{Z}/n\mathbf{Z}$。则集合 $\mathbf{Z}/n\mathbf{Z}$ 对于运算

$$(a \bmod n) \oplus (b \bmod n) = (a+b) \bmod n$$

构成一个加法交换群。

【例 4-5】 设 p 是素数，则集合 $(\mathbf{Z}/p\mathbf{Z})^* = (\mathbf{Z}/p\mathbf{Z}) \setminus \{0\}$ 对于运算

$$(a \bmod p) * (b \bmod p) = (ab) \bmod p$$

构成一个乘法交换群。通常将集合 $\mathbf{Z}/p\mathbf{Z}$ 写成 F_p 或 GF (p)。

设 G 是一个群，如果存在一个元素 g 使得

$$G = \{g^k \mid k \in \mathbf{Z}\}$$

则称 G 是循环群，记为 $G = \langle g \rangle$。这时，g 为群 G 的生成元。

定理 4-1 设 p 是素数，则 F_p 是有限循环群。

设 G 是生成元为 g 的 n 阶循环群，则有如下数学难题：

1）给定整数 a，计算元素 $g^a = h$ 很容易；

2）给定 h，计算整数 x，$0 \leqslant x \leqslant n$，使得 $g^x = h$ 非常困难。

这个难题叫做离散对数问题。

【例 4-6】 $p = 20000000000000002559$ $(\approx 2^{65})$ 是一个素数，$g = 11$ 是 F_p^* 的生成元。

1）对于整数 $a = 20050714$，可以快速计算

$$g^a \equiv 14158167154104328392 \pmod{p}$$

2）但求整数 x，使得

$$g^x \equiv 14158167154104328392 \pmod{p}$$

是非常困难的。

后面要说明椭圆曲线的点对于加法构成一个群，并且存在椭圆曲线的离散对数问题。进一步，可以构建安全的椭圆曲线密码系统。

4.1.2 椭圆曲线上的加法规则

椭圆曲线方程的一般形式为

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

这里，考虑 a_1，a_2，a_3，a_4，a_6 为数域 K 中的元素。

域 K 上的点集为

$$E: \{(x,y) \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{O\}$$

式中，a_1，a_2，a_3，a_4，$a_6 \in K$；$\{O\}$ 为无穷远点，叫做域 K 上的椭圆曲线。

椭圆曲线方程可写成齐次形式

$$Y^2Z + a_1XYZ + a_3Y = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

式中，a_1，a_2，a_3，a_4，$a_6 \in K$。这时，无穷远点 O 就是齐次坐标点 [0:1:0]。

在对域 K 上的椭圆曲线 E 的研究中，通常取如下形式的椭圆曲线方程：

1）当域 K 的特征不为 2、3 时，椭圆曲线方程为

$$y^2 = x^3 + a_4x + a_6$$

2）当域 K 的特征为 2 时，椭圆曲线方程为

$$y^2 + xy = x^3 + a_2x^2 + a_6 \text{ 或 } y^2 + a_3y = x^3 + a_4x + a_6$$

3）当域 K 的特征为 3 时，椭圆曲线方程为

$$y^2 = x^3 + a_2x^2 + a_6 \text{ 或 } y^2 = x^3 + a_4x + a_6$$

1. 椭圆曲线加法规则

设 E 是定义的域 K 上的椭圆曲线

$$E: \{(x,y) \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{O\}$$

定义 E 上的运算法则，记为 $\oplus$：

设 P 和 Q 是 E 上的两个点，L 是过 P 和 Q 的直线（如果 $P = Q$，L 是过 P 点的切线），R 是 L 与曲线 E 相交的第三点。设 L' 是过 R 和 O 的直线，则 $P \oplus Q$ 就是 L' 与 E 相交的第三点。

定理 4-2 E 上的运算法则 $\oplus$ 具有如下性质：

1）如果直线 L 交 E 于点 P，Q 和 R（不必是不同的），则

$$(P \oplus Q) \oplus R = O$$

2）对任意 $P \in E$，$P \oplus O = P$。

3）对任意 P，$Q \in E$，$P \oplus Q = Q \oplus P$。

4）设 $P \in E$，存在一个点，记作 $-P$，使得

$$P \oplus (-P) = O$$

5）对任意 P，Q，$R \in E$，有

$$(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$$

这就是说，E 对于运算法则 $\oplus$ 构成一个交换群。更进一步，如果 E 定义在 K 上，则

$$E(K) := \{(x,y) \in K \times K \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{O\}$$

是 E 的子群。

现在给出定理 4-2 中群运算的具体计算公式。

定理 4-3 设椭圆曲线 E 的一般方程为

$$E: \{(x,y) \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{O\}$$

设 $P_1 = (x_1, y_1)$，$P_2 = (x_2, y_2)$ 是曲线 E 上的两个点，则

1）$-P_1 = (x_1, -y_1 - a_1x_1 - a_3)$

2）如果 $P_3 = (x_3, y_3) = P_1 + P_2 \neq O$，则 x_3、y_3 可以由公式给出

$$\begin{cases} x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - a_1x_3 - y_1 - a_3 \end{cases}$$

其中，

$$\begin{cases}\lambda=\dfrac{y_2-y_1}{x_2-x_1} & x_1\neq x_2\\ \lambda=\dfrac{3x_1^2+2a_2a_1+a_4-a_1y_1}{ay_1+a_1x_1+a_3} & x_1=x_2\end{cases}$$

2. 实数域 R 上椭圆曲线及其运算法则的几何意义

因为实数域 R 的特征不为 2、3，所以实数域 R 上椭圆曲线 E 的方程可设为

$$E:\ y^2=x^3+a_4x+a_6$$

其判断式 $\Delta=-16\ (4a_4^3+27a_6^2)\ \neq 0$。这时，$E$ 在 R 上的运算规则为：

设 $P_1=(x_1,\ y_1)$，$P_2=(x_2,\ y_2)$ 是曲线 E 上的两个点，O 为无穷远点。则

1）$O+P_1=P_1+O$

2）$-P_1=(x_1,\ -y_1)$

3）如果 $P_3=(x_3,\ y_3)\ =P_1+P_2\neq O$

$$\begin{cases}x_3=\lambda^2-x_1-x_2\\ y_3=\lambda\ (x_1-x_3)\ -y_1\end{cases}$$

其中，$\begin{cases}\lambda=\dfrac{y_2-y_1}{x_2-x_1} & x_1\neq x_2\\ \lambda=\dfrac{3x_1^2+a_4}{2y_1} & x_1=x_2\end{cases}$

运算法则的几何意义是：设 $P_1=(x_1,\ y_1)$，$P_2=(x_2,\ y_2)$ 是曲线 E 上的两点，O 为无穷远点，则 $-P_1$ 为过点 P_1 和点 O 的直线 L 与曲线 E 的交点。换句话说，$-P_1$ 是点 P_1 关于 x 轴的对称点，而点 P_1 与点 P_2 的和 $P_1+P_2=P_3=(x_3,\ y_3)$ 是过点 P_1 与点 P_2 的直线 L 与曲线 E 的交点关于 x 轴对称点 $P_3=-R$。

3. 素域 F_p（$p>3$）上的椭圆曲线 E

因为素域 F_p 的特征不是 2、3，所以素域 F_p 上椭圆曲线 E 的方程可设为

$$E:\ y^2=x^3+a_4x+a_6$$

其中，$\Delta=-16\ (4a_4^3+27a_6^2)\ \neq 0$。这时，$E$ 在 F_p 上的运算规则为

设 $P_1=(x_1,\ y_1)$，$P_2=(x_2,\ y_2)$ 是曲线 E 上的两个点，O 为无穷远点，则

1）$O+P_1=P_1+O$

2）$-P_1=(x_1,\ -y_1)$

3）如果 $P_3=(x_3,\ y_3)\ =P_1+P_2\neq O$

$$\begin{cases}x_3=\lambda^2-x_1-x_2\\ y_3=\lambda\ (x_1-x_3)\ -y_1\end{cases}$$

其中，$\begin{cases}\lambda=\dfrac{y_2-y_1}{x_2-x_1} & x_1\neq x_2\\ \lambda=\dfrac{3x_1^2+a_4}{2y_1} & x_1=x_2\end{cases}$

4. 域 F_{2^n}（$n\geqslant 1$）上的椭圆曲线 E

因为 F_{2^n} 的特征为 2，所以域 F_{2^n} 上椭圆曲线 E 的方程可设为

$$E:\ y^2+xy=x^3+a_2x^2+a_6$$

E 在域 F_{2^n} 上的运算规则为：

设 $P_1=(x_1,\ y_1)$，$P_2=(x_2,\ y_2)$ 是曲线 E 上的两个点，O 为无穷远点，则

1）$O+P_1=P_1+O$

2）$-P_1=(x_1,\ -y_1)$

3）如果 $P_3=(x_3,\ y_3)=P_1+P_2\neq O$

$$\begin{cases}x_3=\lambda^2+\lambda+x_1+x_2+a_2\\ y_3=\lambda(x_1+x_3)+x_3+y_1\end{cases}$$

其中，$\begin{cases}\lambda=\dfrac{y_2+y_1}{x_2+x_1} & x_1\neq x_2\\ \lambda=\dfrac{x_1^2+y_1}{x_1} & x_1=x_2\end{cases}$

5. 域 F_{3^n}（$n\geqslant 1$）上的椭圆曲线 E

因为域 F_{3^n} 的特征为 3，所以域 F_{3^n} 上椭圆曲线 E 的方程可设为

$$E:\ y^2=x^3+a_2x^2+a_6$$

E 在域 F_{3^n} 上的运算规则为

设 $P_1=(x_1,\ y_1)$，$P_2=(x_2,\ y_2)$ 是曲线 E 上的两个点，O 为无穷远点，则

1）$O+P_1=P_1+O$

2）$-P_1=(x_1,\ -y_1)$

3）如果 $P_3=(x_3,\ y_3)=P_1+P_2\neq O$

$$\begin{cases}x_3=\lambda^2-x_1-x_2-a_2\\ y_3=\lambda(x_1-x_3)-y_1\end{cases}$$

其中，$\begin{cases}\lambda=\dfrac{y_2-y_1}{x_2-x_1} & x_1\neq x_2\\ \lambda=\dfrac{3x_1^2+2a_2x_2}{2y_1} & x_1=x_2\end{cases}$

4.1.3 建立在有限素数域上的椭圆曲线

有限域上元素的表示方法对椭圆曲线密码系统的可行性、费用和速度都有很大的影响。这个问题很重要，必须强调。但是，有意思的是，某个有限域 F_q 的表示方法并不影响其上的椭圆曲线密码系统的安全性。为了尽可能减小模乘法的运算时间，素数 p 可以是形如 $p=a^k-1$ 的 Mersenne 素数。F_p 的元素为介于 $0\sim p-1$ 之间的整数，用二进制表示。令 $m=[\log_2 p]$，$t=m/32$。

在软件中，将域元素 a 存储于 t 个 32bit 字的数组中。

如前文所述，F_p 域上的椭圆曲线运算公式如下：

$$\begin{cases}x_3=\lambda^2-x_1-x_2\\ y_3=\lambda(x_1-x_3)-y_1\end{cases}$$

其中，$\begin{cases} \lambda = \dfrac{y_2 - y_1}{x_2 - x_1} & x_1 \neq x_2 \\ \lambda = \dfrac{3x_1^2 + a_4}{2y_1} & x_1 = x_2 \end{cases}$

4.1.4 建立在有限二进制域上的椭圆曲线

GF(2^n)域上的元素表示与 GF(p)域不同，最简单的元素表示方法为标准基，有限域 F_{2^m} 可以被看成是 F_2 上的 m 维的向量空间。也就是说，这里存在 F_{2^m} 上包含 m 个元素的集合 $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$，使得对于任意一个 $\alpha \in F_{2^m}$，可以惟一地表示成如下的形式：

$$\alpha = \sum_{i=0}^{m-1} a_i \alpha_i, \qquad a_i \in \{0,1\}$$

可以将元素 α 表示成二进制向量 $(a_0, a_1, \cdots, a_{m-1})$。域上的加法可以由逐比特的 XOR 异或计算来实现。

F_{2^m} 有很多不同的基，下面分别进行介绍。

1. 三项式基（Trinomial bases）

如果 $f(x)$ 是一个 F_2 上的次数为 m 的不可约多项式，那么域 F_{2^m} 可以表示为 F_2 上的次数小于 m 的多项式的集合。其中，多项式的乘法运算需要进行模 $f(x)$ 运算。也就是说，取 α_i 为 x^i，$0 \leqslant i \leqslant m-1$。这样的表现形式成为多项式基表示方法。

三项式基表示方法是多项式基表示方法的一种，其中多项式 $f(x)$ 具有 $f(x) = x^m + x^k + 1$ 的形式。

不论是软件实现，还是硬件实现，这样的表示方法可以简化模 $f(x)$ 运算，有效地提高性能。

2. 最佳正规基（Optimal normal bases）

F_{2^m} 上的一组正规基可以表示为 $\{\beta, \beta^2, \beta^{2^2}, \cdots, \beta^{2^{m-1}}\}$，其中，$\beta \in F_{2^m}$，这样形式的基总是存在的。因为平方运算在 F_{2^m} 域上是线性运算，所以有

$$\alpha^2 = \sum_{i=0}^{m-1} a_i \beta^{2^{i+1}} = \sum_{i=0}^{m-1} a_{i-1} \beta^{2^i} = (a_{m-1}, a_0, \cdots, a_{m-2})$$

因此，一组正规基的表示方法在进行平方运算的时候非常有优势，只要进行一个简单的循环右移就可以实现，这在硬件设计中也是非常好的性能。正规基下的乘法就很复杂了。

所谓的最佳正规基给出了这个问题的最有效的解决方式。正规基的另外一个优势在于 F_{2^m} 中元素的平方根也可以很快地求出来。

3. 利用子域（Using subfields）

如果 $m = lr$，其中，l 是一个小整数，如 8 或者 16，那么有限域 F_{2^m} 可以被看成 F_{2^l} 上的 r 次扩域。如果 $\{\alpha_0, \alpha_1, \cdots, \alpha_{r-1}\}$ 是 F_{2^l} 上的 F_{2^m} 的一组基，那么，对于任意一个元素 $\alpha \in F_{2^m}$ 可以惟一地确定为

$$\alpha = \sum_{i=0}^{r-1} a_i \alpha^i, a_i \in F_{2^l}$$

F_{2^m} 的域乘法涉及到 F_{2^l} 上的一些运算。因为 l 很小，所以 F_{2^l} 上的计算可以运行得非常快。比如，可以预计算对数或者反对数表格来提高运算速度。但是这种方法的缺点在于这些

表格的空间要求很大。

4.2 安全椭圆曲线

当需要使用椭圆曲线密码系统的时候，首先要建立一条有限域上的椭圆曲线。Schoof 的方法被认为是生成椭圆曲线密码系统的椭圆曲线的一种有效的方法。

4.2.1 计算椭圆曲线上点的个数

设 $K=F_q$ 是 $q=p^n$ 元有限域。设 E 是定义在 F_q 上的椭圆曲线，当 $p>3$ 时，椭圆曲线方程可写为

$$y^2=x^3+a_4x+a_6$$

易知，F_q 上椭圆曲线 E 中的点数$\#(E(F_q))\leqslant 2q+1$。事实上，对每个 $x\in F_q$，至多有两个 $y\in F_q$ 使得 $P(x,y)\in E$，再加上无穷远点 O，有$\#(E(F_q))\leqslant 2|F_q|+1=2q+1$。

现在考虑 F_q 上椭圆曲线 E 上的 Frobenius 映射

$$\varphi: F_q\to F_q$$
$$(x,y)\mapsto (x^q,y^q)$$
$$O\mapsto O$$

φ 将 E 上的点映射到 E，且保持群的运算规则。

Frobenius 映射 φ 与其迹在椭圆曲线的研究中起着重要作用。它们由如下方程联系：

$$\varphi^2-[t]\varphi+[q]=[0]$$

也就是，对椭圆曲线 E 上任意点 $P=(x,y)$，有

$$(x^{q^2},y^{q^2})-[t](x^q,y^q)+[q](x,y)=O$$

进一步，有 Hasse 定理

$$\#(E(F_q))=q+1-t$$

对于任意正整数 k，令 $N_k=\#E(F_{q^k})$，有

$$N_k=q^k+1-t_k$$

其中，t_k 满足

$$t_0=2,\ t_1=t,\ t_{k+1}=t_{k+2}-q^kt_{k-1},\ k\geqslant 1$$

又因为 $\varphi_{q^k}=\varphi_q^k$，所以

$$\varphi_q^{2k}-t_k\varphi_q^k+q^k=0$$

因为 t_k 很小，所以计算起来相对简单。

4.2.2 求取椭圆曲线上的点

考虑消息 m（$0\leqslant m\leqslant M$）到椭圆曲线 E 上的对应关系，以及椭圆曲线 E 上点的求取。为简单起见，考虑素域 F_p（$p>3$）上的椭圆曲线 E。

设素域 F_p 上椭圆曲线 E 的方程为

$$E: y^2=x^3+ax+b$$

其中，$16(4a^3+27b^2)\neq 0$。

如何确定 E 上的点 (x_0, y_0)？通常先选定 $x = x_0$，然后求解 $y = y_0$ 使得

$$y^2 = x_0^3 + ax_0 + b \pmod p$$

也就是说，将求取 E 上的点 (x_0, y_0) 转化为求解二次同余方程

$$y^2 = c \pmod p$$

当 c 不能被 p 整除时，方程或者有两个解 $\pm y_0 \bmod p$，或者无解。这样，可以将数字信息 m 与点 P_m (x_m, y_m) 的对应转化为 m 与第一个 x_m 和符号 $\pm$ 的对应。

但并不是对所有的 $x = x_0$ 都有 $y = y_0$，使得 (x_0, y_0) 为 E 上的点，所以需要讨论 m 与 x_m 的对应方法。

假设有一个正整数 k 满足 $p > Mk$，使得对任意的 m，$0 \leqslant m \leqslant M-1$，都存在一个整数 j，$1 \leqslant j \leqslant k$，满足

$$y^2 = (mk + j)^3 + a(mk + j) + b \pmod p$$

则可以建立消息 m，$0 \leqslant m < M$，到椭圆曲线 E 上 P_m (x_m, y_m) 的对应关系。事实上，根据假设条件，对于 m，$0 \leqslant m \leqslant M-1$，可以找到 $x_m = mk + j$，$1 \leqslant j \leqslant k$，使得同余方程

$$y^2 = x_m^3 + ax_m + b \pmod p$$

有解，取其中的一个解为 y_m，则 P_m (x_m, y_m) 是 m 对应的 E 上的点。反过来，对于点 $P_m(x_m, y_m)$，令 $m = [(x_m - 1)/k]$（其中 $[x]$ 表示数 x 的整数部分），则得到点 P_m 与 m 的对应关系。

显然，上述 k 是存在的，具体取多大，要视所选定的曲线而定。

4.2.3 构造一个有限域上的椭圆曲线

一条安全的椭圆曲线，首先要满足以下几个条件：

1）$\#E$ (F_q) 应该能够被一个足够大的素数 n 整除（例如，$n > 2^{160}$）。

2）$\#E$ (F_q) $\neq q$。

3）$n^* q^k - 1$，$1 \leqslant k \leqslant C$，其中，$C$ 是一个足够大的整数，使得寻找 $F_{q^c}^*$ 上的离散对数在计算上不可行。

随机选取参数 a，$b \in F_q$，满足

$$\begin{cases} 4a^3 + 27b^3 \neq 0, & q = p \\ b \neq 0, & q = 2^m \end{cases}$$

然后可以计算 $u = \#(E(F_q))$，且分解 u，判断是否满足安全椭圆曲线的 3 个条件，直到找到为止。针对不同的有限域，这种随机挑选椭圆曲线的方法又可以分为 SEA 和 Satoh 方法。这两种方法针对各自的有限域对随机挑选方法进行了优化，使得在目前椭圆曲线点的计算方面，这两种方法成为应用的主流。

4.3 建立在椭圆曲线上的密码体制

4.3.1 椭圆曲线上的离散对数问题

椭圆曲线密码系统的安全是基于椭圆曲线离散对数问题（ECDLP）的，椭圆曲线离散

对数问题可以描述如下：

给定一条定义于有限域 F_q 上的椭圆曲线 E，n 阶点 $P \in E(F_q)$，

1）对任意整数 l，$0 \leqslant l \leqslant n-1$，计算点 $Q = lP$ 很容易；

2）对于点 Q，求解整数 x，$0 \leqslant x \leqslant n-1$，使得 $xP = Q$ 是很困难的。

4.3.2 椭圆曲线密码算法

将椭圆曲线离散对数问题应用到密码系统中，可以得到椭圆曲线密码算法。

1. 准备工作

选择有限域 F_q、椭圆曲线 E、基点 (x, y)，把明文编码到曲线上的点 (x_m, y_m)，即每个明文都对应一个二维点，选择一个私钥 n。计算公钥 $P = n(x, y)$。

对于 A：私钥 n_A，公钥 $P_A = n_A(x, y)$；

对于 B：私钥 n_B，公钥 $P_B = n_B(x, y)$。

2. 加密

对于任何想加密消息并发送给 A 的人，选择一个随机整数 k，生成密文 $C_m = \{k(x, y), (x_m, y_m) + kP_A\}$。

3. 解密

A 用私钥恢复明文，计算 $n_A(k(x, y))$，计算 $(x_m, y_m) + kP_A - n_A(k(x,y)) = (x_m, y_m) + k(n_A(x,y)) - n_A(k(x,y)) = (x_m, y_m)$

4.3.3 椭圆曲线密码体制的安全性分析

1. 攻击原理

如前文所述，椭圆曲线密码体制的安全是基于椭圆曲线离散对数难解问题的。Pohlig-Hellman 算法将确定 l 的问题归约到确定 l 模 n 的每一个素数因子上，减少计算复杂度。目前，对于 ECDLP 的最好算法就是 Pollard Rho 方法，采用这个方法，经过 $\sqrt{\pi n}/2$ 次椭圆曲线加法以后可以解决 ECDLP 问题。还有人通过将 ECDLP 问题归约到 DLP 问题上来进行算法的改进。

同时，如果有限域 F_p 上的椭圆曲线上椭圆曲线点的个数 $\#E(F_p) = p$，这条曲线就称为素数域不规则（prime-field-anomalous）。理论证明，存在快速有效的方法可以找到这种 $\#E(F_p)$ 的同构，继而解决了 ECDLP 问题。但是这种攻击方法对于其他结构的椭圆曲线并非有效。

这些方法都要求研究者具有相当水平的数学基础。

2. 软件攻击

假设一台每秒能够执行 100 万条指令（MIPS）的计算机每秒钟可以进行 4×10^4 次椭圆曲线加法。也就是说，大约每年可作 2^{40} 次椭圆曲线加法运算。名词“MIPS 年”表示一台运算能力为 MIPS 的计算机一年的计算能力。

表 4-1 表示在不同的 n 下利用 Pollard Rho 方法计算一个离散对数问题所需要的计算能力。举例来说，如果 10000 台计算机中每一台机器都可以达到 1000MIPS 的计算能力，当 $n \approx 2^{160}$，那么单独一个椭圆曲线离散对数问题需要用 85000 年来解决。

3. 硬件攻击

表 4-1　计算能力

域规模/bit	n 的大小	$\sqrt{\pi n}/2$	MIPS 年
163	160	2^{80}	8.5×10^{11}
191	186	2^{93}	7.0×10^{15}
239	234	2^{117}	1.2×10^{23}
359	354	2^{177}	1.3×10^{41}
431	426	2^{213}	9.2×10^{51}

对于一个装备很好的攻击者而言，利用 Pollard Rho 方法并行搜索可以很快地实现对某种特殊用途的硬件进行攻击。根据 1994 年的研究，如果 $n\approx10^{36}\approx2^{120}$，那么耗资 10000 万美元的具有 325000 个处理器的机器破解一个离散对数问题需要花费 35 天。

需要注意的是，以上的分析仅仅用于破解一个端用户的私钥，而真正破解一个系统还要花费同样的时间来破解另一个端用户的私钥。

4.4　椭圆曲线密码体制的实现与应用

4.4.1　椭圆曲线密码算法的软件实现

在椭圆曲线密码体制的软件实现中，最浪费时间的运算莫过于椭圆曲线的点乘运算，即求解 $Q=nP$，也就是群上的重复运算。要实现群上的重复运算，需要实现两个层面的运算：一个是椭圆曲线群的运算；另一个是椭圆曲线点的运算。

1. F_p 域上的椭圆曲线群的运算

在这个域上的群运算可以分解为：加法、减法、乘法、求逆运算。F_p 上的元素经常表示为整数模 p 的形式，因此，还需要进行模运算，如同 RSA 或者 DSA 系统中的模运算一样。也就是说，原有 RSA 或者 DSA 软件库中的部分函数仍然可以取出来作为 F_p 域上椭圆曲线实现的一部分。

在这些运算中，求逆运算需要格外注意。在大部分公钥密码系统中，模逆（即在有限域中的求逆运算）并不重要。但是在椭圆曲线密码系统中，如果仍然按照以前的方法进行模逆运算，这个运算将成为系统性能的主要瓶颈。一般来讲，可以通过转变椭圆曲线点的表示方法来减少系统中需要求逆的次数。

2. F_p 域上椭圆曲线点的运算

椭圆曲线点上的运算包括加法和点乘。

在加法的运算过程中需要求解 λ，也就是经过点 P_1、P_2 的直线的斜率。其中涉及到了大整数的逆运算，这个运算成为椭圆曲线加法最主要的时间消耗。为了提高加法的运算性能，需要将椭圆曲线点的表示方法从二维坐标（X，Y）转换为三维坐标（X，Y，Z）。

这里存在 3 种坐标转换的方式：

1）标准投影坐标：$(X:Y:Z)\leftrightarrow(X/Z,\ Y/Z)$

相应的椭圆曲线公式为：$Y^2Z=X^3-3XZ^2+bZ^3$

2）Jacobian 投影坐标：$(X:Y:Z)\leftrightarrow(X/Z^2,\ Y/Z^3)$

相应的椭圆曲线公式为：$Y^2=X^3-3XZ^4+bZ^6$

3）Chudnovsky-Jacobian 投影坐标：$(X:Y:Z)\leftrightarrow(X:Y:Z:Z^2:Z^3)$

其中，$(X:Y:Z)$ 是 Jacobian 投影坐标。这样，在这几种坐标下，加法和倍乘算法的步

骤如表 4-2 所示。

表 4-2　不同坐标下计算椭圆曲线点的加法和倍乘算法的步骤

倍乘		一般加法		混合坐标	
$2A \rightarrow A$	$1I, 2M, 2S$	$A+A \rightarrow A$	$1I, 2M, 1S$	$J+A \rightarrow J$	M, S
$2P \rightarrow P$	$7M, 3S$	$P+P \rightarrow P$	$12M, 2S$	$J+C \rightarrow J$	M, S
$2J \rightarrow J$	$4M, 4S$	$J+J \rightarrow J$	M, S	$C+A \rightarrow C$	M, S
$2C \rightarrow C$	$5M, 4S$	$C+C \rightarrow C$	M, S		

其中，A 表示二维坐标下的运算，P 表示标准投影坐标下的运算，J 表示 Jacobian 投影坐标下的运算，C 表示 Choudnovsky-Jacobian 投影坐标下的运算，I 表示求逆运算，M 表示乘法运算，S 表示平方运算。

在椭圆曲线点的点乘运算中，可以分为两种情况：一种是点固定的情况；一种是点未知的情况。可以对固定点的点乘算法进行一些预计算来提高运算的性能。

下面列出了一些 F_p 域上椭圆曲线可能用到的算法。

(1) 算法 1　模加运算

输入：模数 p，a，$b \in [0, p-1]$。

输出：$c = (a+b) \bmod p$

1) $c_0 = \text{Add}(a_0, b_0)$

2) For i from 1 to $t-1$ do: $c_i \leftarrow \text{Add\_with\_carry}(a_i, b_i)$

3) If the carry bit is set, then subtract p from c

4) If $c \geqslant p$ then $c \leftarrow c-p$

5) Return (c)

其中 Add (　) 函数为大整数加法运算，Add_with_carry (　) 函数为大整数带进位加法运算。

(2) 算法 2　模逆运算

输入：素数 p，$a \in [1, p-1]$

输出：$a^{-1} \pmod p$

1) $u \leftarrow a$，$v \leftarrow p$，$A \leftarrow 1$，$C \leftarrow 0$

2) While $u \neq 0$ do

a. while u is even do:

$u \leftarrow u/2$. If A is even then $A \leftarrow A/2$; else $A \leftarrow (A+p)/2$

b. while v is even do:

$v \leftarrow v/2$. If C is even then $C \leftarrow C/2$; else $C \leftarrow (C+p)/2$

c. if $u \geqslant v$ then: $u \leftarrow u-v$, $A \leftarrow A-C$; else $v \leftarrow v-u$, $C \leftarrow C-A$

3) return C mod p

(3) 算法 3　平方

输入：整数 $a \in [0, p-1]$

输出：$c = a^2$

1) For i from 0 to $2t-1$ do: $c_i - 0$

2) For i from 0 to $t-1$ do

a. $(uv) \leftarrow c_{2i} + a_i^2$

b. $c_{2i} \leftarrow v$, $C1 \leftarrow u$, $C2 \leftarrow 0$

c. For j from $i+1$ to $t-1$ do

$(uv) \leftarrow c_{i+j} + a_i a_j + C1$, $C1 \leftarrow u$

$(uv) \leftarrow v + a_i a_j + C2$, $c_{i+j} \leftarrow v$, $C2 \leftarrow u$

d. $(uv) \leftarrow C1 + C2$, $C2 \leftarrow u$

e. $(uv) \leftarrow C1 + C2$, $C2 \leftarrow u$

f. $(uv) \leftarrow c_{i+r} + v$, $c_{i+r} \leftarrow v$

3) Rehurn (c)

(4) 算法 4　未知点的点乘

输入：$k = (k_{m-1}, \cdots, k_1, k_0)_2$, $P \in E(F_p)$

输出：kP

1) $Q \leftarrow O$

2) For i from $m-1$ downto 0 do

a. $Q \leftarrow 2Q$

b. If $k_i = 1$ then $Q \leftarrow Q + P$

3) Return (Q)

(5) 算法 5　固定点的点乘

输入：Window width w, $d = \lceil m/w \rceil$, $k = (k_{d-1}, \cdots, k_1, k_0)_{2^w}$, $P \in E(F_p)$

输出：kP

1) Precomputation. Compute $P_i = 2^{wi}P$, $0 \leqslant i \leqslant d-1$

2) $A \rightarrow O$, $B \rightarrow O$

3) For j from $2^w - 1$ downto 1 do

a. For each i for which $k_i = j$ do: $B \leftarrow B + P_i$ {Add Q_j to B}

b. $A \leftarrow A + B$

4) Return (A)

3. F_{2^m}域上的椭圆曲线群的运算

如前文所述，F_{2^m}域上的椭圆曲线在域表示的基底上有多种表示方案，如三项式基、多项式基和最佳正规基等。对于每一种基底，在实现方面各有其优势和不足，因此有相应的域运算算法与之相对应。本书仅以多项式基为例进行讲解。

设$f(X) = x^m + r(x)$是一个m次的不可约二进制多项式。也就是说，这个多项式的最高次数为m，而每一项系数$\in \{0, 1\}$。F_{2^m}上的元素为次数最多为$m-1$的二进制多项式。在这个域上进行加法、乘法等运算的时候，都需要模$f(x)$。一个域元素$a = (a_{m-1}, \cdots, a_2, a_1, a_0)$，可以用长度为$m$的二进制向量$\boldsymbol{a} = (a_{m-1}, \cdots, a_2, a_1, a_0)$来表示。令$t = [m/32]$，$s = 32t - m$。在软件实现中，利用32bit的字数组来存储域元素。例如，$A = (A[t-1], \cdots, A[2], A[1], A[0])$，其中，$A[0]$最右边的比特是$a_0$，$A[t-1]$最左边的$s$个比特一般不用，设为0。

值得一提的是，域元素的加法可以逐比特地进行，因此只需要t次字运算。

4. F_{2^m}域上的椭圆曲线点的运算

在上文中，针对 F_p 域上的点乘，将乘法根据椭圆曲线点是否已知进行分类。同样，在 F_{2^m} 域中也进行这样的分类，原因在于，对于已知点的乘法可以进行算法改进。

下面列出了一些 F_{2^m} 域上的椭圆曲线点可能用到的算法。

(1) 算法 6　多项式加法

输入：Binary polynomials, $a(x)$ and $b(x)$ of degree at most $m-1$

输出：$c(x) = a(x)\ b(x) \bmod f(x)$

1) If $a_0 = 1$ then $c \leftarrow b$; else $c \leftarrow 0$

2) For i from 1 to $m-1$ do

a. $b \leftarrow b \cdot x \bmod f(x)$

b. If $a_i = 1$ then $c \leftarrow c + b$

3) Return (c)

(2) 算法 7　模多项式运算

输入：Abinary polynomial $c(x)$ of degree at most $2m-2$

输出：$c(x) \bmod f(x)$

1) Precomputation. Compute $u_k(x) = x^k r(x)$, $0 \leqslant k \leqslant 31$

2) For i from $2m-2$ downto m do

If $c_i = 1$ then

Let $j = \lfloor (i-m)/32 \rfloor$ and $k = (i-m) - 32j$

Add $u_k(x)$ to $C\{j\}$

3) Return $((C[t-1], \cdots, C[1], C[0]))$

(3) 算法 8　多项式表示的未知点的点乘

输入：$k = (k_{r-1}, \cdots, k_1, k_0)_2, P \in E(F_{2^m})$

输出：kP

1) $Q \leftarrow O$

2) For i from $t-1$ downto 0 do

a. $Q \leftarrow 2Q$

b. If $k_i = 1$ then $Q \leftarrow Q + P$

3) Return (Q)

(4) 算法 9　多项式表示的固定点的点乘

输入：window width w, $d = \lceil t/w \rceil$, $k = (k_{d-1}, \cdots, k_1, k_0)_{2^w}$, $P \in E(F_{2^m})$

输出：kP

1) Precomputation. Compute $P_i = 2^{wi}P$, $0 \leqslant i \leqslant d-1$

2) $A \leftarrow O$, $B \leftarrow O$

3) For j from $2^w - 1$ downto 1 do

a. For each i for which $k_i = j$ do: $B \leftarrow B + P_i$ {Add Q_j to B}

b. $A \leftarrow A + B$

4) Return (A)

4.4.2　椭圆曲线密码算法的硬件实现

当在高速硬件平台如 PC 上使用软件形式的公钥密码系统时，计算时间和程序的大小并

不是值得特别关注的事情。但是，当在那些具有严格的硬件限制的环境中运行公钥密码系统的时候，它们有限的计算能力、很小的 ROM/RAM 容量突出了公钥密码系统对系统环境的要求。这时需要一种具有小的数据和计算复杂度的公钥密码系统，这种系统才能在受限的硬件环境中正常使用。

椭圆曲线密码系统 ECC 是目前满足这些要求的一种公钥密码系统。在同等安全要求下，椭圆曲线密码系统密钥的长度比传统的公钥密码系统要短。例如，安全参数为 160bit 的 ECC 可以提供与安全参数为 1024bit 的 RSA 公钥密码系统同等的安全性能。更小的密钥长度可以带来更小的系统参数、更小的带宽要求、更快的实现和更低的电力消耗。当然，在软件实现中，这些优点同样是让人振奋的。

4.4.3 椭圆曲线 Diffie-Hellman 密钥交换

1994 年，Scheidler、Buchmann、Williams 利用一个非群的结构，也就是实二次数域，来实现 Diffie-Hellman 密钥交换（ECDH）协议。

1）用户 A、B 已经商定好的参数：E 表示椭圆曲线，n 表示椭圆曲线群上基点 P 的阶，F_q 表示有限域。

2）用户 A 随机选择 $d_A \in [1, n-1]$，计算 $Q_A = d_A P$，发送 Q_A 给 B。

3）用户 B 随机选择 $d_B \in [1, n-1]$，计算 $Q_B = d_B P$，发送 Q_B 给 A。

4）用户 B 计算 $K_B = d_B Q_A$，用户 A 计算 $K_A = d_A Q_B$。

5）得到的密钥为：$K = d_A d_B P = K_A = K_B$。

4.4.4 椭圆曲线数字签名算法

椭圆曲线数字签名算法（ECDSA）是对美国政府数字签名算法（DSA）在椭圆曲线上的模拟。ECDSA 已经为 ANSI 标准、ISO 标准和 IEEEP1363 标准所接纳。

1. ECDSA 密钥生成

E 是有限域 F_q 上的椭圆曲线，P 是椭圆曲线 E（F_q）上阶为 n 的一点，这些都是系统参数。为了描述简便，假设 q 就是一个素数，尽管在实际应用中 q 还可以是素数的幂，但是这种假设不失一般性。

那么，对于每一个端用户 A 来说，密钥生成的步骤是：

1）随机选择整数 $d \in [1, n-1]$；

2）计算 $Q = dP$；

3）用户 A 的公钥为 Q，私钥为 d。

2. ECDSA 签名生成

为了对消息 m 进行签名，用户 A 需要：

1）随机选择整数 $k \in [1, n-1]$；

2）计算 $kP = (x_1, y_1)$，$r = x_1 \bmod n$；

3）其中，x_1 需要介于 0 和 $q-1$ 之间。如果 $r=0$，退回到第一步；

4）计算 $k^{-1} \bmod n$；

5）计算 $s = k^{-1} \{h(m) + dr\} \bmod n$，其中，$h$ 是安全哈希函数（SHA-1）。如果 $s=0$，

退回到第一步；

6）消息 m 签名就是这样的一对整数（r，s）。

3. ECDSA 签名验证

为了验证用户 A 关于 m 的签名（r，s），用户 B 需要：

1）得到用户 A 的公钥 Q 的可信副本；

2）验证 r 和 s 是介于 1 和 $n-1$ 之间的整数；

3）计算 $w=s^{-1}\bmod n$ 和 h（m）；

4）计算 $u_1=h$（m）$w\bmod n$ 和 $u_2=rw\bmod n$；

5）计算 $u_1P+u_2Q=$（x_0，y_0）和 $v=x_0\bmod n$；

6）当且仅当 $v=r$ 时，接受这个签名。

DSA 和 ECDSA 之间最大的区别在于 r 的生成。DSA 通过随机取得元素 $\alpha^k\bmod p$，然后将其进行模 p 运算来达到同样的目的。

在 DSA 中，q 是一个 160bit 的 $p-1$ 的素因子，α 是域 F_p^* 上的 q 阶元素。

ECDSA 是对点 kP 的 x 坐标进行模 n 运算。为了达到和 DSA 一样的安全强度，参数 n 需要有 160bit 长，这样的话，DSA 和 ECDSA 具有相同的比特长度 320bit。

如果通信的两端都仅仅固定有限域 F_q 而没有系统参数，那么每个端系统都需要有自己的椭圆曲线 E 和点 $P\in E$（F_q）。在这种情况下，椭圆曲线 E 的定义、点 P、点 P 的阶 n 都需要包含在端系统的公钥中。

如果有限域 F_q 是固定的，那么就可以根据这个具体的有限域对硬件和软件实现进行优化。同时，在一个有限域 F_q 上，有很多椭圆曲线可供选择。

4.5 小结

椭圆曲线密码算法是目前最有活力的公钥密码算法之一，其优越的安全性能和广泛的应用环境使得越来越多的机构采用椭圆曲线密码算法作为其主要的公钥算法。本章从椭圆曲线的数学原理开始介绍，然后对椭圆曲线上的计算法则进行了描述，并且对椭圆曲线密码体制作了较为全面的分析，包括椭圆曲线离散对数问题、密码体制安全性以及加密、解密算法。最后，本章给出了椭圆曲线密码算法的软件实现的部分算法和对硬件算法的分析，并描述了椭圆曲线 Diffie-Hellman 密钥交换（ECDH）协议和椭圆曲线 ECDSA 签名算法。

4.6 习题

1. 计算椭圆曲线点：

考虑定义在 Z_{23} 上的椭圆曲线 E：$y^2=x^3+x+1$。E（Z_{23}）是循环群，其中的一个生成元是 $P=$（0，1），求所有 E（Z_{23}）上的点。

2. 计算椭圆曲线点：

考虑定义在 Z_{23} 上的椭圆曲线 E：$y^2+xy=x^3+x^2+1$，其不可约多项式是 $f(x)=x^3+x+1$，E（F_{23}）是循环群，其中一个生成元是 $P=$（α，α^5），求所有 E（F_{23}）上的点。

第5章 量子密码

20世纪，科学家发现经典物理学是用来描述宏观世界的强大工具，但是对于微观世界却毫无用处。宏观世界中的确定性无法描述本质的随机微观粒子，而且微观世界中，海森伯格（Heisenberg）的测不准原理（Uncertainty Principle）加强了测量中“精确”的局限性。

量子信息进程是一个新兴的研究领域，人们研究在信息处理中利用量子系统和量子法则的可能性。经过近几年的不断努力，发现经典信息处理中被认为的一些难题在量子系统是可以被轻松解决的，如果量子计算机制造出来，很多公钥密码系统方案将完全无用。

当经典密码学利用各种各样的数学方法来限制窃听者得到加密信息的时候，量子机器中的信息则由物理法则来保护。在经典密码学中，并不能保证信息达到所谓的绝对安全。而应用海森伯格测不准原理和量子纠缠特性的安全通信系统，也就是通常所说的“量子密码学”，提供密钥协商或者交换密钥时的绝对安全。

5.1 量子比特的基本属性

比特（bit）是经典计算和经典信息的基本概念，量子信息同样建立在一个类似的概念上：量子比特（Quantum Bit 或 Qubit）。

量子比特的属性是建立在量子力学的基础上的。量子信息技术主要来源于几个量子力学方面的定理和规则。

1. 测不准原理

假设一个粒子的动量为 p，位置为 x。在量子世界中，不可能同时精确地测量 p 和 x。不确定性总是伴随着每次测量发生，即使是在非常完美的实验中。不确定性不是独立的，存在如下的关系：

$$\mathrm{d}p\mathrm{d}x > \frac{h}{2\pi}$$

其中，h 是普朗克常数。

也就是说，如果某物的位置 x 被精确地确定，那么对此物动量的限定将是非常脆弱的，反之亦然。没有人可以同时得到任意精度的物体位置和动量。

测不准原理带来很多奇特效应。在量子世界中，不能100%地确定粒子的位置，只能用概率的方法表示。例如，只能说一个原子以99%的概率存在于某个地方，而仍有1%的概率在其他的地方，可能是宇宙中的任何一个地方。

2. 不可克隆原理

对于非正交的两个量子状态不可克隆。这与经典计算机中的电子比特不同。在经典计算机中，信息可以被任意精确地复制，甚至在传统纸质媒体中，信息也可以很容易地被复制。在量子世界中，只有正交的两个量子状态才可以被复制，对于非正交的量子状态，不可复制。

5.1.1 量子叠加性

量子比特同样具有两种可能的状态：$|0>$和$|1>$。“$|\ >$”称为 Dirac 记号，在量子力学中表示状态。与比特不同的是，量子比特的状态并不是非 0 即 1 的，而是一个连续的状态空间。量子比特的状态可以是这些状态的线性组合，表示如下：

$$|\psi> = \alpha|0> + \beta|1> \tag{5-1}$$

其中，α、β是复数，$|0>$、$|1>$是二维复空间中的一组正交基，而且α、β满足$|\alpha|^2 + |\beta|^2 = 1$。每个量子比特可以被看作是这个二维复空间中的单位向量。

但是α、β的值不能得到，在测量的过程中，只能得到量子比特的一部分信息。也就是说，有$|\alpha|^2$的概率取 0，有$|\beta|^2$的概率取 1。测量结束以后，量子比特也会改变其原先的状态，呈现测量结果所显示的状态：非 0 即 1。

单量子比特还可以利用几何方法来表示，即

$$\left|\varphi\right> = \cos\frac{\theta}{2}\left|0\right> + e^{i\varphi}\sin\frac{\theta}{2}\left|1\right> \tag{5-2}$$

其中，θ、φ是实数。

这样，就可以用球面来表示单量子比特，这个球面叫做 Bloch 球面，如图 5-1 所示。

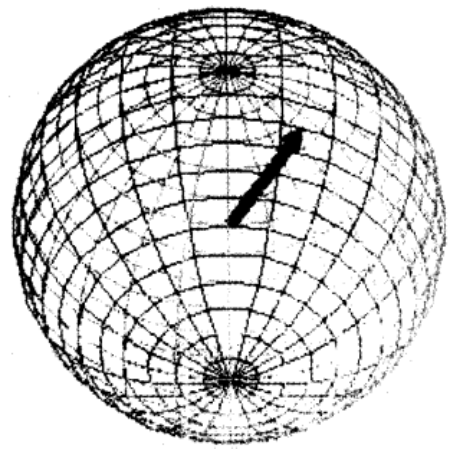


图 5-1 Bloch 球表示量子比特

5.1.2 单量子比特变换

如 5.1.1 节所述，单量子比特可以用二维复空间的单位矢量表示，如同二维复空间存在多种基底一样，单量子比特也可以采用不同的二维基底来表示。

例如，对于状态$\left|+\right> = \frac{1}{\sqrt{2}}\left|0\right> + \frac{1}{\sqrt{2}}\left|1\right>$和状态$\left|-\right> = \frac{1}{\sqrt{2}}\left|0\right> - \frac{1}{\sqrt{2}}\left|1\right>$，可以用这两个状态作为二维基底表示单量子比特：

$$\begin{aligned}|\varphi> &= \alpha|0> + \beta|1> \\ &= \alpha\frac{|+> + |->}{\sqrt{2}} + \beta\frac{|+> - |->}{\sqrt{2}} \\ &= \frac{\alpha+\beta}{\sqrt{2}}\left|+\right> + \frac{\alpha-\beta}{\sqrt{2}}\left|-\right>\end{aligned} \tag{5-3}$$

当二维基底正交时，可以保证其归一化条件$\left|\alpha\right|^2 + \left|\beta\right|^2 = 1$，从而能够得到测量结果。也就是说，对于$|\varphi> = \alpha|0> + \beta|1>$，相对基底$|0>$和$|1>$测量以后，以$|\alpha|^2$的概率得到结果$|0>$，以$|\beta|^2$的概率得到结果$|1>$，而相同的$|\varphi>$，采用不同的正交基底$|+>$和$|->$，$\left|\varphi> = \frac{\alpha+\beta}{\sqrt{2}}\right|+> + \frac{\alpha-\beta}{\sqrt{2}}\left|->$，相对基底$|+>$和$|->$测量以后，以$\frac{\left|\alpha+\beta\right|^2}{2} = \frac{1}{2}$的概率得到结果$\left|+\right>$，以$\frac{1}{2}$的概率得到结果$\left|-\right>$。

5.1.3 量子逻辑门

经典计算机是由很多逻辑门电路（如与非门、异或门、非门等）构成的。同样，量子计算机也可以通过一些基本量子逻辑门来构造。

1. 单比特量子逻辑门

在经典逻辑非门中，其输出是输入的非，即 0 变 1，1 变 0。但是量子比特的状态是线性的，对于某个量子比特

$$\alpha|0>+\beta|1>$$

如果通过量子非门，则应该变成

$$\beta|0>+\alpha|1>$$

这是一个线性变换，可以通过矩阵来表示，即量子非门

$$\boldsymbol{X}\equiv\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$$

对于量子比特 $\alpha|0>+\beta|1>$，通过非门 $\boldsymbol{X}$ 以后得到

$$\boldsymbol{X}\begin{pmatrix}\alpha\\ \beta\end{pmatrix}=\begin{pmatrix}\beta\\ \alpha\end{pmatrix}$$

由于每个量子比特都可以表示成一个单位矢量，因此，要求用于表示这个量子非门的矩阵必须是酉阵。

除了上文描述的非门以外，量子比特非门还有其他形式。

$$\boldsymbol{Z}\equiv\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix},\qquad \boldsymbol{H}\equiv\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$$

对于量子比特 $\alpha|0>+\beta|1>$，通过 $\boldsymbol{Z}$ 门以后得到

$$\boldsymbol{Z}\begin{pmatrix}\alpha\\ \beta\end{pmatrix}=\begin{pmatrix}\alpha\\ -\beta\end{pmatrix}$$

对于量子比特 $\alpha|0>+\beta|1>$，通过 $\boldsymbol{H}$ 门以后得到

$$\boldsymbol{H}\begin{pmatrix}\alpha\\ \beta\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix},$$

值得注意的是，$\boldsymbol{H}^2=\boldsymbol{I}$。也就是说，连续两次通过 $\boldsymbol{H}$ 门，相当于回到最初的状态。$\boldsymbol{H}$ 门也叫做 Hadamard 门。同样的，$\boldsymbol{Z}$ 门和 $\boldsymbol{H}$ 门也是酉阵。

2. 多量子比特门

在经典计算机电路中，存在与门、或门、异或门、与非门等。在量子计算中，也同样存在多量子比特门。

（1）受控非门

受控非门如图 5-2 所示。

受控非门的输入是图 5-2 中左边两个量子比特 $|A>$ 和 $|B>$，输出是右边的 $|A>$ 和 $|B\oplus A>$。输入 $|A>$ 为控制量子比特，输入 $|B>$ 为受控量子比特。当控制量子比特为 1 时，受控量子比特取非。也就是说，若输入为 $|00>$，则输出为 $|00>$；若输入为 $|01>$，则输出为 $|01>$；若输入为 $|10>$，则输出为 $|11>$；若输入为 $|11>$，则输出为 $|10>$。如果用矩阵来表示则是：

$$U_{CNOT}=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}$$

这也是一个酉阵。

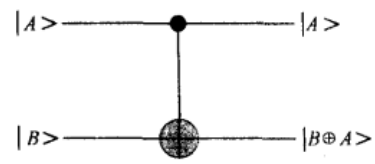


图 5-2　受控非门

（2）Toffoli 门

如图 5-3 所示，Toffoli 门有两个控制量子比特 $|A>$ 和 $|B>$，有一个受控量子比特 $|C>$ 作为输入，输出的结果是 $|A>$、$|B>$ 和 $|C\oplus AB>$。

当 $|C>$ 为 $|1>$ 时，可以得到 $|1\oplus AB>$，也就是与非门；当 $|A>$ 为 $|1>$，$|C>$ 为 $|0>$ 时，可以得到 $|0\oplus B>$。这样，就实现了对量子比特的输出。

5.1.4　Bell 理论与 EPR 纠缠态

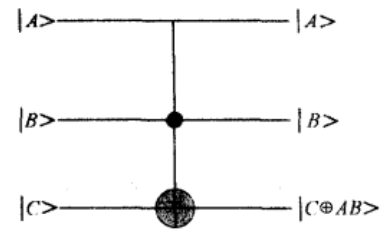


图 5-3　Toffoli 门

Bell 理论产生于 EPR 佯谬。尼尔斯·波尔于 1927 年提出了互补性原理（Complementary Principle），试图对量子力学进行解释。但是爱因斯坦对此表示不理解，他对量子力学中用概率进行量子描述存在疑问，认为“上帝不掷骰子”。因此，就互补性理论，也就是量子力学理论，两位诺贝尔奖获得者之间展开了热烈的讨论。

1935 年，爱因斯坦及其同事发表了论文《量子力学对物理现实的描述完备吗?》（Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?），认为成功的物理理论应该同时满足两个条件：

1）理论正确；

2）对这个理论的描述是完备的。

对于现实中的每一个物理量，在物理理论中都应该有一个对应物。他们认为如果在没有任何对系统的干扰的情况下，应该能够肯定地预测一个物理量的值，还会有一个物理现实中的元素对应于这个物理量。爱因斯坦利用一个反例，证明了量子力学的理论描述是不完备的。在这篇论文中，爱因斯坦提出了一个假设，也就是后来被称为 Locality Principle 的原理：量子间的传输速度不超过光速，不存在超距作用。

1964 年，Bell 发表论文，指出任何企图支持爱因斯坦的 Locality Principle 的理论都会和量子力学发生冲突。这就是 Bell 理论。

EPR 纠缠态也叫做 Bell 态，形式如下：

$$|\beta_{00}>=\frac{|00>+|11>}{\sqrt{2}} \tag{5-4}$$

$$|\beta_{01}>=\frac{|01>+|10>}{\sqrt{2}} \tag{5-5}$$

$$|\beta_{10}>=\frac{|00>-|11>}{\sqrt{2}} \tag{5-6}$$

$$|\beta_{11}> = \frac{|01> - |10>}{\sqrt{2}} \tag{5-7}$$

这 4 种状态中的每一种状态都是由两个量子比特构成的。利用纠缠态，可以在没有信道的情况下实现通信。

如果 Alice 和 Bob 在通信前曾经见过面，将一个 EPR 纠缠态中的两个量子分别拿好，若 Alice 需要向 Bob 发送一个比特的信息，那么 Alice 只需让那个信息比特与她手上的纠缠态中的一个量子比特相互作用，然后测量她的两个比特，把结果告诉 Bob，Bob 根据 Alice 的测量结果，测量他手上的量子比特，就可以知道正确的结果。

在这个过程中，并没有违背量子不可克隆原理，最初承载信息的量子比特因为与纠缠态量子比特相互作用而改变了，只有 Bob 手中的那个纠缠态量子比特还保有信息。而且，Alice 和 Bob 必须通过传统的信道进行通信，将 Alice 的测量结果告诉 Bob，这样 Bob 才能根据 Alice 的测量结果选择合适的测量方法，也就是滤波器，才能恢复出信息。因此，这种被称为“量子隐形传态”的传输也没有超过光速。

EPR 纠缠态的这种特点在 1991 年被 Ekert 利用，设计出 E91 量子密钥分发协议。

5.2 量子密码基本原理

量子密码是一种物理密码，它利用量子操作对信息做加密和解密处理，利用量子力学的基本属性获得理论安全性。20 世纪 60 年代末期，物理学家 S. Wiesner 产生了在密码学中采用测不准原理的想法（尽管很晚他才把自己的想法发表出来）。当时，S. Wiesner 写了一篇题为《共轭编码》的论文，他的想法之一就是利用带有极性的光子流来传送两个信息，而这两个消息在接收者的选择之下只有一个是可读的。这个被称做“多路技术”的想法和很多年以后被重新使用的“one-out-of-two oblivious transfer”技术非常相似，它甚至比 Rabin 的“不经意传输”概念提前了 10 多年！Wiesner 的这篇论文开创了量子密码的先河，在密码学史上具有划时代的意义。遗憾的是，这篇论文当时没能获得发表，直到 1983 年才得以面世，使得量子密码学的出现推迟了 10 年。70 年代末期，Wiesner 的想法被 C. H. Bennett 和 G. Brassard 在 Crypto’82 上的论文再次提出。在 1984 年，Bennett 和 Brassard 终于提出了著名的量子密钥分发的概念。这个概念的提出意味着量子密码的真正开始。同年，Brassard 又提出了量子掷币协议（Quantum Coin Flipping Protocol），后来发展成量子比特承诺。从此，量子密码学引起了国际密码学界和物理学界的高度重视。在 IBM 的 Watson 实验室成功地建立了量子密钥交换信道的工作模型之后，量子密码学引起了人们越来越大的关注。

1991 年，牛津大学的 Ekert 提出了 E91 协议，1992 年，Bennett 指出只用两个非正交态即可实现量子密码通信，并提出了 B92 协议。自此，量子密码通信的三大主流方案已基本形成。

5.2.1 量子密码信息理论

量子密码系统就是使得发送方用量子信道与接收方共享秘密信息，而未经授权的第三方无法窃取信息的量子系统。一个量子密码系统由量子信源、量子信道、经典信道、发送方、

接收方等部分组成。

量子信源是量子信道之源。量子信源主要提供信道中所需的量子态。在量子信源中选定一个量子码组后，量子码组中各个量子态出现的几率可以是各种情况，但量子密码系统中所选用的量子态都是等几率出现的。需要指出的是，量子信源与经典信源是完全不同的。因为在量子信源中，当量子码组确定后，量子信道也就确定了。

量子信道用于获取秘密信息。所谓量子信道就是将量子比特从一端送往另一端的量子信息通道。量子信道可以有很多种形式，如自由空间（验证 BB84 协议的第一个实验所用的量子信道为自由空间）、光纤等。

在经典信息论中，信道只是将信息从发送方传输到接收方，然而在量子信息论中，信道中的信息传递方式是很重要的，信息的传输受制于信息载体。

5.2.2 对敌手的检测理论

这里讨论敌手对量子密码系统的作用。

假设 Alice 准备一个量子系统 Q，初态为 $\rho^{(Q)}$。设在量子系统 Q 中有 n 个信号量子态 $\rho_k^{(Q)}$，$k=1, 2, \cdots, n$，则

$$\rho^{(Q)} = \sum_{k=1}^{n} p_k \rho_k^{(Q)}$$

其中，ρ_k 为每个态出现的概率。

Alice 通过一个有噪声量子信道将量子系统传输给 Bob，在传输中受到环境 E 的作用，环境的作用可以用超算符 $E^{(Q)}$ 表示。为不失一般性，设环境的初态为纯态 $|0_E>$，则环境作用后的量子系统为：

$$\rho^{(Q')} = \sum_{k=1}^{n} p_k E^{(Q)}(\rho^{(Q)})$$

于是信道中 Alice 传输给 Bob 的信息量由下式确定：

$$I_{ab} \leqslant S(\rho^{(Q')}) = \sum_{k=1}^{n} p_k S(\rho_k^{(Q')}) = \chi^{(Q')}$$

Eve 截取信息量为

$$I_{eab} \leqslant S(\rho^{(E')}) = \sum_{k=1}^{n} p_k S(\rho_k^{(E')}) = \chi^{(E')}$$

定义量子保密度 P，则：

$$P = \frac{I_{ab} - I_{eab}}{I_{ab}^{max}}$$

其中，$I_{ab}^{max} = \ln N$。

由上面的式子可知，P 的取值范围为 $-1 \leqslant P \leqslant 1$。

当 $P<0$ 时，Eve 的作用引起 Alice 和 Bob 之间的信息量减少，从而 Alice 和 Bob 能够检测出 Eve；当 $P>0$ 时，Eve 的作用引起 Alice 和 Bob 间的信息量增加，增加的信息量将背叛 Eve，从而 Alice 和 Bob 能够检测出 Eve；当 $P=0$ 时，Eve 的作用能逃脱 Alice 和 Bob 间的检测，但要求 Eve 能正确地选取与 Alice 或 Bob 的量子态相同的态，而在 Alice 和 Bob 随机选取量子态时，这种情况发生的概率极低。此外，当 Alice 和 Bob 选取共轭量子态时，Eve 的作

用也能使得 $P=0$。因此满足条件 $P\neq 0$ 的量子密码系统为完善保密系统。

在量子密码系统中，上述条件是很容易实现的，所以，目前量子密码学中所提出的量子密钥分发协议都具有无条件安全性。

5.2.3 保密加强理论

密码系统一旦建立，敌手就会利用各种方法对其进行分析和破译。我们应分析敌手的可能行为，并在密码系统的设计中加以防范，以增强系统的安全性。目前，对量子密码系统的攻击方法有个体攻击、集体攻击和联合攻击。

个体攻击就是攻击者独立地为每一个截获到的量子态设置一个探测器，然后测量每一个探测器中的粒子，从而获取信息。

集体攻击就是用一个试探粒子与通信者的量子态序列相互作用，之后对试探粒子测量，以此获取信息。

联合攻击就是以通信者间的量子态为基础产生量子纠缠态，通过纠缠态的性质获取最大信息。

5.3 量子密钥分发

目前至少存在3种主要用于量子密钥分发的量子密码系统，它们是：

1）基于4个量子状态来协商密钥的BB84协议的量子密码系统；

2）基于相位编码的B92协议的量子密码系统；

3）基于纠缠光子间的纠缠特性的EPR协议的量子密码系统。

5.3.1 BB84协议

BB84协议是第一个量子密码通信协议，它由Benntt和Brassard于1984年提出。BB84实现了两方通过共享的量子信道和公开的经典信道建立一个对称加密用的密钥，也就是会话密钥。目前，有很多研究小组在实现和实验这样的量子密钥分发方案，这个协议已经在实际光纤和自由空间实验中得到了验证。一些公司甚至已经开始在实际中使用这些成果。

（1）编码

首先，需要肯定的是，Alice想要发送给Bob的密钥实际上是一串比特。每一个比特的值将用光子的某种属性来表示。在BB84协议中，有4种极性来表示比特的值：水平、竖直和对角。从图形上看，这4种极性可以表示为，↔、↕、↖↘和↗↙。Alice和Bob商定、↔、↖↘表示0，↕、↗↙表示1。一个滤镜用来区分↔和↕光子，而另一个滤镜用来区分↖↘和↗↙光子。因此，每一个滤镜都可以读出一个光子到底是0还是1。

（2）工作流程

Alice为密钥上的每一个比特，从一个光子的两种可能的表现形式中随机地选择一种极性，然后通过量子信道将它们发送给Bob。在每个光子传输中，Bob选择一个滤镜来读光子，可能是用来读竖直/水平光子的，也有可能是用来读对角光子的。然后，Bob告诉Alice他的选择，Alice告诉他这些选择中，哪些是正确的。那些被选择正确的滤镜读出的比特确定了密钥的基础。然后为了从这些被正确读出的比特中抽取真正的密钥，他们开始检查传输

中是否有窃听者的存在或者不法传输。

简单地说，方案之所以安全是因为如果窃听者 Eve 试图读传输中的光子，那么在她读光子的次数中，平均至少有一半会改变光子的极性。如果没有 Eve 的窃听行为，则光子因传输而极性改变的个数非常少。这样，Alice 和 Bob 就可以识别出她的存在。换言之，自然法则保证了 Eve 会以几乎确定的概率泄露她自己或者得不到关于密钥的任何信息。窃听者不被检测出来并且得到大量信息的概率是非常小的。

（3）协议

1）对于每一个密钥比特，Alice 发送一个光子，其极性为随机挑选的。她记录下这些极性方向。

2）对于每一个接收到的光子，Bob 随机选择两个滤镜中的一个，写下他的选择，并且记录下他的过滤结果。

3）等到所有的光子传输完毕，Bob 通过一个传统的不安全信道，如电话，告诉 Alice 他选用的滤镜的类型。

4）Alice 告诉 Bob 他的选择中有哪些是正确的滤镜。

5）Alice 和 Bob 现在知道他们各自拥有的比特串中，哪些比特是相同的，即那些 Bob 使用正确滤镜读出的比特。这些相同的比特的一个子集将会是他们的会话密钥。

6）最后，Alice 和 Bob 检查他们持有的这些相同的比特串。在这个步骤中，将会用到差错校验码（Error Correcting Codes），其中的一些比特将会被扔掉，留下来的就是他们的密钥。

图 5-4 为 BB84 协议实验原理图。Alice 的信号源提供量子状态，通过对称光束滤波器以后进入一个双臂 Mach-Zehnder 干涉仪。在其中一臂中，Alice 用相移器 A 对量子进行相移，在另一端，Bob 用相移器 B 对接收到的量子比特进行相移。这里相移器就是滤波器。

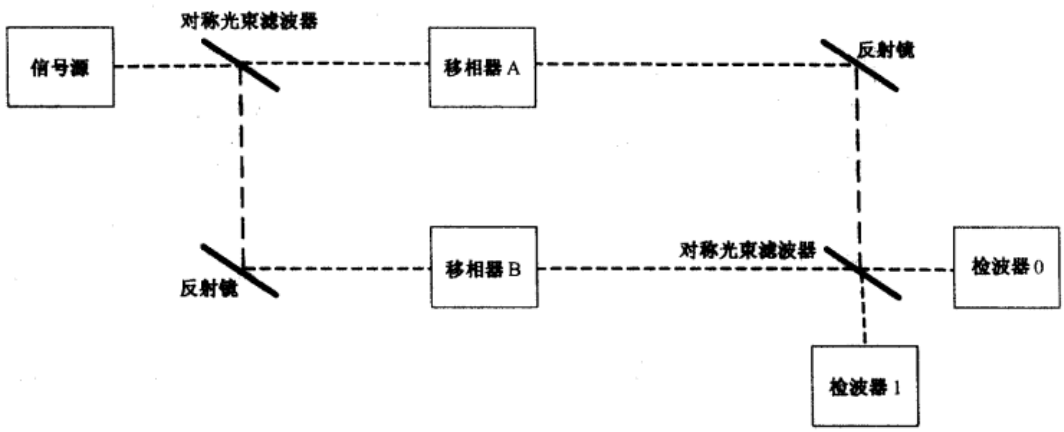


图 5-4　BB84 协议实验原理图

5.3.2　B92 协议

对于一种测量，如果没有办法干扰两个非正交状态，那么测量同样不能生成任何可能区

分两个非正交态的信息。非 EPR 密钥分发方案的安全性即来源于这样的事实。这个事实同样使得可以只采用两个非正交状态来实现密钥分发协议。

（1）编码

设 $|u_0\rangle$ 和 $|u_1\rangle$ 是量子比特两个可区分的非正交状态。$P_0 = 1 - |u_1\rangle\langle u_1|$，$P_1 = 1 - |u_0\rangle\langle u_0|$ 是 $|u_1\rangle$ 和 $|u_0\rangle$ 的正交子空间上的射影算子。因此，P_0 湮灭了 $|u_0\rangle$，P_1 湮灭了 $|u_1\rangle$。

（2）协议

1）Alice 准备一个任意的二进制量子比特序列并发送给 Bob。其中，状态 $|u_0\rangle$ 表示 0，状态 $|u_1\rangle$ 表示 1。

2）对于每个量子比特，Bob 独立、随机地选择 P_0、P_1 两个测量中的一个进行测量。

3）Bob 公开告诉 Alice 他测量得到正结果的量子比特而不是测量方法。双方商定废弃掉其他的量子比特。

如果传输过程中没有监听，那么 Alice 发送 $|u_0\rangle$ 且 Bob 用 P_0 测量以及 Alice 发送 $|u_1\rangle$ 且 Bob 用 P_1 测量的量子比特将与分式 $\frac{1 - |\langle u_0|u_1\rangle|^2}{2}$ 相关。

在密钥建立以前，Alice 和 Bob 仍然要消耗一些保留下来的量子比特，用来验证他们手中的量子比特是否完全一致。同样，这也证明了是否有监听的存在。若存在监听，则必然在传输过程中干扰 $|u_0\rangle$ 或 $|u_1\rangle$，影响后面的测量结果。

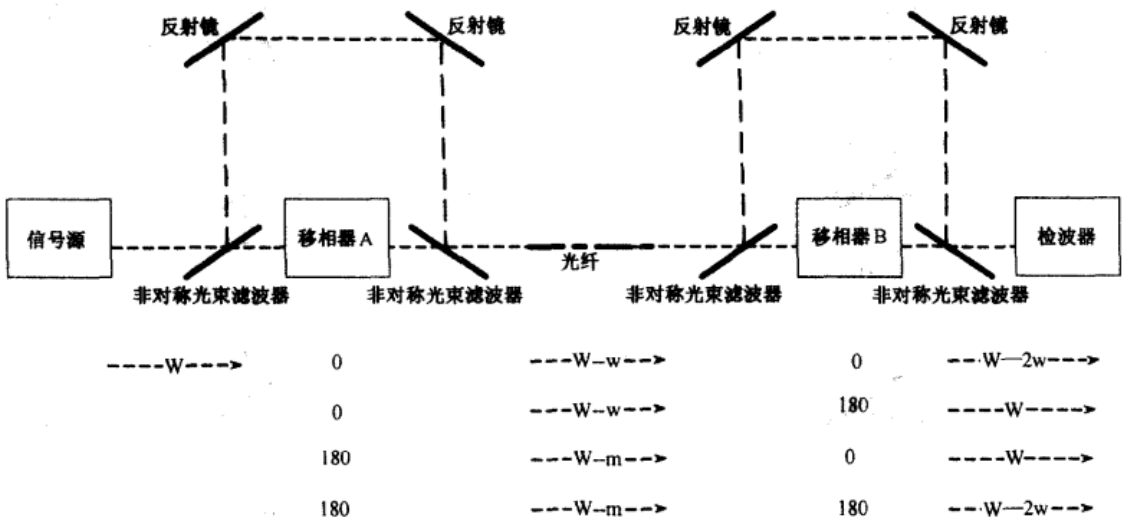


图 5-5　B92 协议实验原理图

（3）实验分析

在图 5-5 中，两个非正交状态 $|u_0\rangle$ 和 $|u_1\rangle$ 是两股微弱的相干脉冲，两个脉冲的相位不同，同时还伴随一个较强的参考脉冲。

实验从图的左边开始。Alice 用一系列非对称光束滤波器和反射镜（半干扰器）来同时分开初始的光束：一束微弱的强度 $u<1$ 期望光子强度的信号脉冲后面跟随一束较亮的强度 $M>1$ 期望光子的参考脉冲。信号脉冲通过移相器 A 后，相移 0°或者 180°，这样就可以进行 0、1 编码了。然后发射光束到一根单模光纤中。较亮的参考脉冲不经过相移，但是延迟一段固定的时间 Δt 后才发射进入同样的单模光纤中。

在接收端，Bob 用与 Alice 相同的半干扰器来分开光束，按照相同的比率分成一束微弱的光束和一束较强的光束。同样，微弱的光束通过相移器 B 随机相移 0°或者 180°，这个过程与 Alice 的相移独立。同时，参考光束延迟 Δt。两部分光束进入检波器后变成相干光束。

进入检波器的光束由 3 个脉冲组成，它们之间延时 Δt 的倍数。第一个脉冲，因为 Alice 和 Bob 的测量而削弱，但是没有任何时延。第二个脉冲包含了重要的密钥信息，其中有 Alice 延迟 Bob 削弱的光束，也有 Alice 削弱 Bob 延迟的光束。如果 Alice 和 Bob 的相移是相等的，就会出现相长干涉并且重叠的脉冲会以概率 $4\mu Tq$ 期望光子产生一个计数，其中，T 是光纤的传输参数，q 是检波器的量子功率。如果 Alice 和 Bob 的相移不等，叠加脉冲强度会比较低，在理想状况下是 0。最后的脉冲是一个较亮的脉冲，Alice 和 Bob 都对其做过延迟，但都没有对其削弱。在叠加脉冲到达 Δt 的时间后，第三个脉冲进入检波器。这束脉冲不含有任何信息，主要用于确认参考脉冲的到达。这样可以防止监听。

这种方案尽管比 BB84 协议少用了 2 个状态，但是因为只有大约一半的测量得到正结果，效率减低了一半。

5.3.3 E91 协议

E91 协议是建立在 EPR 纠缠态上的协议。它与 BB84 协议类似，主要的区别在于 Bob 手上的量子不是由 Alice 通过光纤发送过来的，而是通过 Alice 和 Bob 之间的一个量子比特源，这个发射源分别向 Alice 和 Bob 的方向发射 EPR 对，每边发射 EPR 对中的一个，这样 Alice 和 Bob 就有很多用于隐形传输的 EPR 对了。这种方案克服了量子无法在一根光纤中远距离传输的困难。

因为 E91 协议是 Ekert 基于 EPR 佯谬提出的一种量子密钥分发协议，因此，这个协议也叫做 EPR 协议。

5.4 小结

本章介绍了不同于经典信息技术的量子信息技术。对量子信息的基本概念的描述，包括量子基本属性，单量子比特变换、量子门、量子 EPR 纠缠态等方面。并且，对著名的量子密钥分发协议 BB84、B92、E91 协议作了介绍。通过学习本章的内容，读者对量子密码技术能有一个初步的了解。

5.5 习题

1. 给出 Toffoli 门的矩阵表示。

2. 给出图 5-6 所示量子线路的结果。

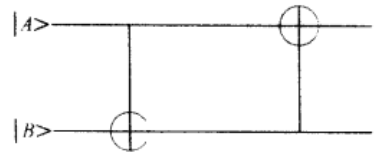


图 5-6　量子线路图

第6章 混沌密码

Robert A. J. Matthews 在 1989 年首次将混沌用于密码学研究，并提出了一种基于变形 Logistic 映射的混沌流密码方案。从此混沌密码学作为密码学的一个分支得到了广泛的研究。美国海军实验室研究人员 Pecora 和 Carroll 首次利用驱动-响应法实现了两个混沌系统的同步，这一突破性的研究成果为混沌理论在通信领域中的应用开辟了道路。随着混沌理论研究的不断发展，国内外许多学者对基于混沌理论的加密方法设计及其安全性进行了广泛而深入的探讨，并逐渐形成了混沌密码学这一新的研究分支。特别是 1997 年以后，为了将混沌密码学进一步实用化，学者们提出了许多新的数字化混沌密码方法，从而掀起了数字混沌密码学研究的高潮。

现代密码按加密方式进行分类，可以分为分组密码和序列密码（流密码）；按密钥管理的方式来划分，可分为公开密钥算法和传统密钥算法（对称算法）。因为混沌系统具有的宽频谱、类随机特性、对结构参数及初始状态的极端敏感性等一系列性质，使得混沌成为密码学研究的一个重要领域。混沌系统产生的序列从严格意义上来讲属于流密码，但它与传统的流密码又有区别。从目前对混沌密码的研究状况来看，它有可能成为一类具有广阔应用前景的加密方式。

6.1 混沌学基本原理

可以用相空间中的轨道来表示经典力学中的各种运动形式。如果运动方程不含随机项，那它所描述的就是一种确定性的运动。混沌运动是确定性系统中局限于有限相空间的高度不稳定的运动。所谓轨道高度不稳定，是指临近的轨道随着时间的变化会呈指数方式分离。这种由系统参数或初始条件所造成的不同的演化轨道随时间的平均发散率可用 Lyapunov 指数来测度。这里 Lyapunov 指数就是描述两个靠近的初值所产生的轨道，随时间的推移按指数方式分离的参量。

在一维动力系统 $x_{n+1} = F(x_n)$ 中，初始两点迭代后是互相分离还是靠拢，关键取决于导数 $\left|\frac{dF}{dx}\right|$ 的值。如果 $\left|\frac{dF}{dx}\right| > 1$，则迭代的结果使得两点分开；如果 $\left|\frac{dF}{dx}\right| < 1$，则迭代使得两点靠拢。但是在不断的迭代过程中，$\left|\frac{dF}{dx}\right|$ 的值也随之变化，使得迭代过程中两点时而分离时而靠拢。为了从整体上描述相邻两个状态分离的情况，必须对时间取平均，因此 Lyapunov 指数可写为：

$$\lambda = \lim \frac{1}{n} \sum_{i=1}^{n-1} \ln \left| \frac{dF(x)}{dx} \right|_{x=x_i} \tag{6-1}$$

Lyapunov 指数的值越大表明轨道的平均发散速率越快。工程上常用最大 Lyapunov 指数来测度轨道的平均发散速率，而且它也可以作为系统是否是混沌系统的判据。当 Lyapunov

指数大于0时，系统就进入了混沌状态。

由于混沌系统的不稳定性，系统的长时间行为会显示出某种混乱性。混沌是一种貌似无规则的运动，不需附加任何随机因素亦可出现类似随机的行为，其最大特点在于系统的演化对初始条件十分敏感。从长期意义上讲，系统的未来行为是不可预测的。混沌不是简单的无序，而是没有明显的周期和对称，但却是具有丰富的内部层次的有序结构。混沌运动在相空间的轨迹具有复杂的拉伸、折叠和收缩的结构，但每一条轨迹不自我重复，且局限于有限集合，这称为奇怪吸引子。

美国著名气象学家洛伦兹在研究天气预报问题时，首先发现了确定性方程中出现的混沌现象。他将所研究的对流模式简化后，得到如下的三维常微分方程组：

$$\begin{cases} \dot{x} = -\sigma(x - y) \\ \dot{y} = -xz + rx - y \\ \dot{z} = xy - bz \end{cases} \tag{6-2}$$

该方程组就是著名的Lorenz方程。固定参数 $\sigma = 10$，$b = 8/3$，随着 r 的变化，r 在大于24.74后系统进入“混沌区”，对混沌区内的参数 r 值，有时定常态是混沌解，即吸引子是奇怪的，有时也可能出现稳定的周期解，在 $r = 28$ 时，动力系统呈混沌态。

图6-1是求解Lorenz方程所得的混沌序列，左边是对 x 采样所得的时间序列，右边是对 y 采样所得的时间序列。可以看出，这些序列有明显的类随机特性，系统的未来行为是无法预测的。

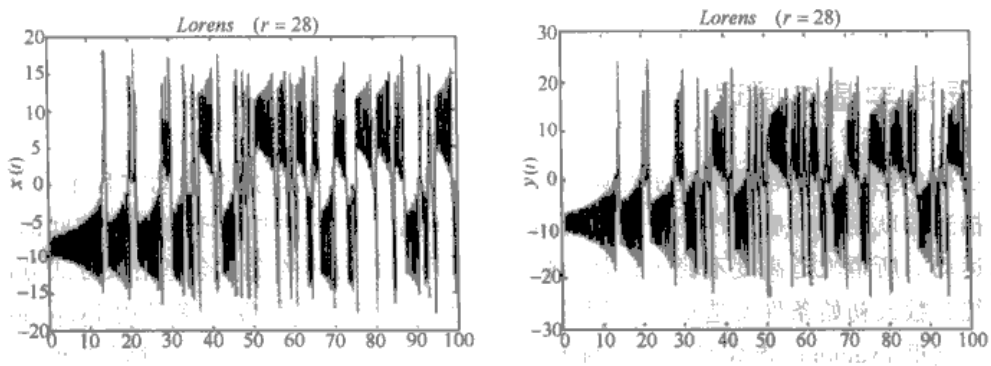


图6-1 Lorenz方程采样序列

图6-2是系统吸引子在 $x-z$ 平面、$x-y$ 平面上的投影，它们具有极其复杂的自相似结构。奇怪吸引子也可作为混沌运动的判据，具有奇怪吸引子的运动就是混沌运动。

与其他复杂现象相比，混沌运动有着自己的特征：

1）遍历性。混沌运动在其混沌吸引域内是各态历经的，即在有限时间内混沌轨道经过混沌区内每一个状态点。

2）有界性。混沌是有界的，它的运动轨迹始终局限于一个确定的区域，这个区域称为混沌吸引域，从整体上来说混沌系统是稳定的。

3）内随机性。混沌的内随机性实际就是它的不可预测性，对初值的敏感性造就了它的这一性质。

4）标度性。标度性指混沌运动是无序中的有序态，它仍是确定性迭代。

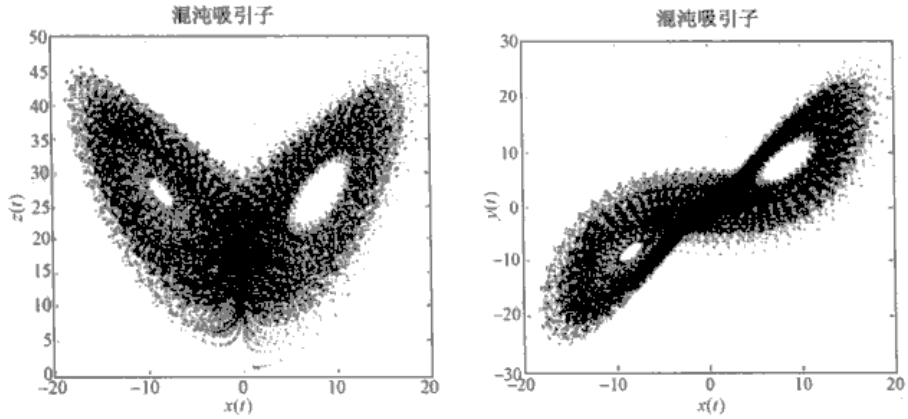


图 6-2　混沌吸引子

5）分维性。分维性表示混沌运动状态具有多叶、多层结构，且叶层越分越细，表现为无限层次的自相似结构。

6）普适性。普适性是指不同系统在趋向混沌态时所表现出来的某些共同特征，它不依具体的系统方程或参数而变。普适性是混沌内在规律的一种体现。

6.1.1　控制混沌原理

如何利用及控制混沌是当今自然科学基础研究的热点问题之一，对这一问题的研究具有重要的理论和实际意义。这是因为：

1）人们希望能够找到一些方法来控制系统中的混沌和分岔行为，以消除或减少某些实际系统中的混沌和分岔现象的发生。

2）混沌在某些环境下是有用的，当系统处在混沌状态时，它包含各种各样失稳的周期、准周期运动。如果能够找到一些方法把系统从混沌运动状态变为所希望的周期、准周期运动状态，那就为混沌的利用提供了一些方法。因此，控制混沌的主要目的就是消除已有的混沌运动，或者降低混沌运动程度。从原则上讲，通过对实际系统进行修改或施加控制总会影响混沌运动的生存条件，从而可设法消除或者抑制混沌运动。但是，这些控制方法尚未利用混沌的内在动力学特性，实现中往往要对原系统作较大的修改或输入较大的控制能量。因此，由于混沌系统的特殊性，使得系统的预测估计和控制都比较困难。

严格地说，线性系统只是非线性系统的一种特例，而非线性系统才是最一般的系统。混沌是某些非线性系统所特有的一种运动形态。因此，如何控制和利用混沌在工程上也具有十分重要的意义。

混沌运动有 3 个特点：

1）混沌轨道是一个非周期轨道，它在变量空间中占据了比任何周期轨道都大得多的范围。确切地说，就是在变量空间中混沌吸引子具有比任何周期吸引子大的维数，而在混沌运动中系统在混沌吸引子上各态历经。这样，人们就可以在整个混沌吸引子的范围内来实行控制操作和选择控制目标态，这使混沌控制具有很大的灵活性。

2）混沌运动又有初值敏感性，任何邻近轨道之间的距离随时间的演化会以指数形式迅速发散，这导致了混沌运动长时间趋势行为的不可预见性。但是，这一特性在混沌控制中有

可能成为一个重要优点，即对系统施加极小的影响就可以使运动产生重大变化，这使人们有可能在混沌条件下用精心选择的微小信号来灵活而有效地控制系统的运动结果，这一可能在周期条件下是绝对不能想象的。

3）混沌运动的内涵极为丰富，这是任何一个周期轨道都无法相比的优点。在任何稳定的混沌状态中镶嵌着无穷多个周期轨道，这些周期轨道为混沌控制目标态的选择提供了极为丰富的内容。下面介绍几种重要的混沌控制方法。

1. OGY 方法

美国马里兰大学物理学家 Ott、Grebogi 和 Yorke 于 1990 年基于有无穷多的不稳定周期轨道嵌入在混沌吸引子中这一事实，提出了一种利用混沌内在特征的控制策略，简称为 OGY 方法。其思想实质是：无穷多不稳定周期轨道稠密地分布在混沌吸引子闭包上，因此有可能迫使混沌态变为稳定的周期运动状态。连续对系统的参数施加小扰动，使在无穷多不稳定周期轨道中所期望的那个不稳定周期轨道稳定化，即系统做所要求的有规律性的周期运动，达到控制混沌的目的。具体做法是：首先由低周期起步，逐步确定出混沌吸引子闭包上的一些不稳定周期轨道，之后对系统施加小扰动，以使某个选定的不稳定周期轨道稳定化。

考虑可调参数时间连续 D 维系统。其动力学方程为：

$$\mathrm{d}\boldsymbol{x}/\mathrm{d}t = \boldsymbol{F}(\boldsymbol{x}, p)$$

这里 $\boldsymbol{x}$ 为 D 维态矢量，p 为可调参数。如果不知道描述系统的动力学方程，系统的消息仅由某个测量过程得到，则可用系统相空间 D 中的标量函数 z 来表示该测量过程。如果系统在 t 时刻的状态是 $\boldsymbol{y}(t) \in D$，那么所测得时间序列的标量函数为 $z(t) = z(\boldsymbol{y}(t))$。利用时间序列重构技术，采用延迟时间为 T，嵌入维为 M 的延迟坐标，可重构出 M 维延迟矢量：

$$\boldsymbol{x}(t) = (z(t), z(t-T), \cdots, z(t-(M-1)T) \in R^M$$

一般选择 $\boldsymbol{x}(t)$ 的一个分量为常数，如 $[\boldsymbol{x}(t_n)]_M \equiv z(t_n - MT) = C$ 为试验截面。这种方法给出了截面上的点列 $\boldsymbol{\xi}_n \in R^{M-1}$ 以及它们之间的映像 $\boldsymbol{\xi}_{n+1} = \boldsymbol{f}(\boldsymbol{\xi}_n, p)$。在此截面上，时间连续的周期轨道表现为有限点集的时间离散轨道，与一般嵌入要求不同，在此实现点集嵌入时，嵌入维 M 只要与该点坐标维数相同就足够了，因此 $M = D - 1$。为了简化，取 $D = 3$，$M = 2$。对序列 $\boldsymbol{\xi}_n$ 可确定混沌吸引子的许多不稳定周期轨道，还要选出适合的不稳定周期轨道，并把它稳定化。为简化，假定选择一个周期轨道，它是截面映像不动点。令 $\boldsymbol{\xi}(p_0) = \boldsymbol{\xi}_F(p_0) = 0$ 是所期望的不动点。假设参数 p 在 p_0 附近变化。如果令 $p_0 = 0$，则 p 的变化范围可取为 $p^* > p > -p^*$，当 p 从 $p_0 = 0$ 变至小量 $\bar{p}$ 时，不动点坐标 $\boldsymbol{\xi}_F(p_0)$ 从 0 移到邻近点 $\boldsymbol{\xi}_p(\bar{p})$。

在 $\boldsymbol{\xi}_F(p_0) = 0$ 附近对映像作线性近似为：

$$[\boldsymbol{\xi}_{n+1} - \boldsymbol{\xi}_F(p)] \approx \boldsymbol{M}[\boldsymbol{\xi}_n - \boldsymbol{\xi}_F(p)]$$

其中，$\boldsymbol{M}$ 是 2×2 矩阵。令不动点 $\boldsymbol{\xi}_F(p) \approx p\boldsymbol{g}$，式中 $\boldsymbol{g} \equiv \left.\dfrac{\partial \boldsymbol{\xi}_F(p)}{\partial p}\right|_{p_0=0} \approx \bar{p}^{-1}\boldsymbol{\xi}_F(\bar{p})$ 是由序列确定的矢量。进而可把改变参数后的线性映像式子化为：

$$\boldsymbol{\xi}_{n+1} \approx p_n\boldsymbol{g} + (\lambda_u \boldsymbol{e}_u \boldsymbol{f}_u + \lambda_s \boldsymbol{e}_s \boldsymbol{f}_s)(\boldsymbol{\xi}_n - p_n\boldsymbol{g})$$

其中，$\boldsymbol{e}_u$ 和 $\boldsymbol{e}_s$ 是不稳定与稳定流形方向上的单位矢量；λ_u 和 λ_s 是不动点处的不稳定与稳定本征值（$|\lambda_u| > 1 > |\lambda_s|$）。$\boldsymbol{g}$、$\boldsymbol{e}_u$、$\boldsymbol{e}_s$、$\lambda_u$ 和 λ_s 均可由嵌入序列得到。$\boldsymbol{f}_s$、$\boldsymbol{f}_u$ 是协变基矢。$\boldsymbol{\xi}_n$ 落在期望不动点 $\boldsymbol{\xi}_F = 0$ 附近。此时选择 p_n 使 $\boldsymbol{\xi}_{n+1}$ 落在 $\boldsymbol{\xi} = 0$ 处的稳定流形上，即不稳定流

形方向 f_u 与轨道方向 ξ_{n+1} 垂直：

$$f_u \xi_{n+1} = 0$$

当 ξ_n 足够小时，可得到确定 p_n 的公式，这里记为 C_n：

$$C_n = \lambda_u (\lambda_{u-1})^{-1} (\xi_n f_u)/(g f_u) \equiv C\xi_n$$

当上式右边的绝对值小于 p^* 时，$p_n = C_n$ 即为所求，反之置 $p_n = 0$，即 p_n 取为：

$$p_n = \begin{cases} C_n, & |C_n| < p^* \\ 0, & |C_n| \geqslant p^* \end{cases}$$

这样，要求 $gf_u \neq 0$，且仅当 ξ_n 落入 $|\xi_n^u| < \xi^*$ 的狭区中才能施加扰动（$\xi_n^u = f_u \xi_n$）。令 $C_n = p^*$，$\xi_n = \xi^*$，从而确定 p^*：

$$p^* = \xi^* / |(1 - \lambda_u^{-1}) g f_u|$$

2. 混沌的连续控制方法

一般来说，控制混沌的目的是镇定混沌吸引子中嵌入的不稳定周期轨道。上面介绍的 OGY 法是一种十分有效的控制方法。但是 OGY 方法有其局限性：

1）一般只适用于可以构造 Poincare 映射的连续系统或者离散系统。

2）参数变化范围较小，环境噪声及系统参数偶然跳变都会使系统偏离周期轨道，尤其是 OGY 法需要混沌系统在不稳定周期轨道附近的准确模型来调节控制参数，应付不稳定流形。

当采用 OGY 法控制连续系统中的混沌现象时，因为控制信号是脉冲式的，所以难以避免控制间隙外部噪声产生的严重干扰。1992 年，德国物理学家 K. Pyragas 提出了两种联系混沌控制方法：外力反馈控制法和延迟自反馈控制法。其基本思想都是考虑混沌系统的输出信号与输入信号之间的自反馈耦合。它们的原理图如图 6-3 所示。

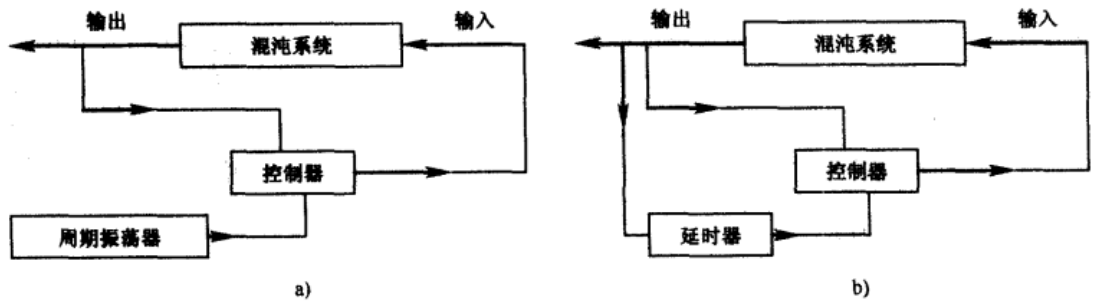


图 6-3　混沌联系控制方法原理图

a）外力反馈控制法示意图　b）延迟自反馈控制法示意图

3. 外力反馈控制法

假设非线性连续系统的动力学特性可以由一个非线性常微分方程组来描述。系统的具体动力学模型并不要求知道，但通过试验可以测得系统的某个输出量。外力反馈控制法的特点是，当无扰动的系统存在混沌时，从外部给系统输入一个用以与系统的输出量进行比较的强迫信号，同时给出控制信号作为对系统的一种微扰。当然，前提是无微扰时的系统必须存在混沌吸引子，只有这样才有可控制的无穷多的周期轨道或非周期轨道。利用延迟坐标的标准方法，可以从混沌吸引子内大量的不同周期的不稳定轨道中提取一个标量信号，于是可以得到相应于不同的不稳定周期态的周期信号。为了达到控制的目的，这一技术原则上要能产生

无穷多的周期信号。

对于连续动力系统：

$$\begin{cases}\dot{x} = p(x,y)\\ \dot{y} = q(x,y) + f(t)\end{cases}$$

y 是某一个可测量的量，$f(t)$ 是输入控制量。用 $y_i(t)$ 代表吸引子中周期为 T_i 的不稳定的周期轨道，令控制信号为：

$$f(t) = k[y_i(t) - y(t)] \quad (k > 0)$$

当 $y(t) = y_i(t)$ 时，$f(t) = 0$。也就是说，当系统沿着以 T_i 为周期的轨道运行时，系统中的这条轨道没有受到 $f(t)$ 的影响。$y_i(t)$ 通常由对吸引子中不稳定的周期轨道的重构得到。

4. 混沌的自适应控制方法

H. K. Qammar 等人于 1993 年根据自适应控制理论，提出了针对混沌的间接自适应控制方法，如图 6-4 所示。

对于如下一个 N 维混沌动力系统：

$$\dot{X} = F(X,U,t)$$

其中，$X = \{x_1,x_2,\cdots,x_N\}$ 代表系统的状态变量，$U = \{u_1,u_2,\cdots,u_m\}$ 代表系统参数。

控制时另加一个调节 U 的动力方程：

$$\dot{U} = EG(X - X_s)$$

其中，X_s 代表目标状态，E 是控制刚度，有 $\lim\limits_{x\to X_s} G(X - X_s) = 0$。

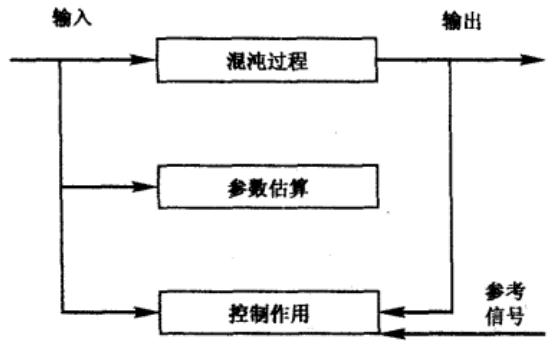


图 6-4　混沌的间接自适应控制

这种间接自适应控制有两种方式，一种叫做提前一步控制法，另一种叫做权重提前一步法。前一种方法要求驱动输入一步到位，因此经常会导致输入太大，无法实现稳定控制。后一种方法把受控的动力学行为与所预估参数的模型方程联立起来，构成一个封闭回路系统，而该封闭系统并不依赖于混沌过程的参数，即模型方程中的参数不必对应于混沌过程的实际参数。这种间接自适应控制能达到较好的效果。

6.1.2　混沌同步原理

考虑如下两个非线性动力系统：

驱动系统：

$$\dot{x} = F(t,x) \tag{6-3}$$

响应系统：

$$\dot{y} = F'(t,y) + U(t,x,y) \tag{6-4}$$

其中，$x = \{x_1(t),x_2(t),\cdots,x_n(t)\}^T \in \mathbf{R}^n$ 为驱动系统的状态变量；$y = \{y_1(t),y_2(t),\cdots,y_n(t)\}^T \in \mathbf{R}^n$ 为相应系统的状态变量 F，$F':[\mathbf{R}_+ \times \mathbf{R}^n] \to \mathbf{R}^n$ 为非线性映射，$\mathbf{R}_+$ 为非负实数集。

令 $x(t;t_0,x_0)$、$y(t;t_0,y_0)$ 为式 (6-3) 和式 (6-4) 在初始条件 (t_0,x_0) 的解，如果存在 $D(t_0) \subseteq \mathbf{R}^n$，$\forall x_0,y_0 \in D(t_0)$，当 $t\to\infty$ 时，有：

$$\lim | x(t;t_0,x_0) - y(t;t_0,y_0) | = 0$$

成立，则称系统（6-3）和（6-4）同步。$D(t_0)$ 为同步区域，$U:[\mathbf{R}_+ \times \mathbf{R}^n \times \mathbf{R}^n] \to \mathbf{R}^n$ 为同步控制量。如果 $F = F'$，则称为自同步。

令 $e = x - y$ 为误差向量，则由系统（6-3）和（6-4）得到其同步的误差系统为：

$$\dot{e} = F(t,x) - F'(t,y) - U(t,x,y)$$

由 e 在原点处的稳定性，可得到同步的稳定性。

以上同步的定义不仅适合混沌同步，而且也适合非混沌同步；既适合自治系统的同步，也适合非自治系统的同步，甚至可以推广到超混沌同步。在混沌同步中，主要采用 Lyapunov 稳定性定理和 Lyapunov 指数来判断同步的稳定性。

下面介绍两种常用的同步方法。

1. 自适应同步法

自适应控制是现代控制理论的重要组成部分，目前已在军事、航天等领域得到了广泛的应用。John 和 Amritker 于 1994 年提出了一种自适应控制来实现混沌同步的方法。该方法可用来控制混沌系统的相空间轨道与所期望的不稳定轨道达到同步。

一般来说，采用自适应同步方法有两个前提条件：

1）系统至少有一个或多个参数可以得到。

2）对于所期望的轨道，这些参数值是已知的。受控参数的变化依赖于两个因素：①系统的状态变量与期望轨道的相应变量的差值；②受控参数值与期望轨道相应的参数值之差。

设一个 n 维自治系统 $\dot{x} = f(x,p)$，其中，$x = \{x_1, x_2, \cdots, x_n\}^T$ 是系统状态变量，$p = \{p_1, p_2, \cdots, p_m\}^T$ 是系统的参数，$f:\mathbf{R}^{n\times m} \to \mathbf{R}^n$ 为矢量函数。

对已经给定的系统 $\dot{y} = f(y,p^*)$，其中，$y = \{y_1, y_2, \cdots, y_n\}^T$ 是系统状态变量，$p^* = \{p_1, p_2, \cdots, p_m\}^T$ 是系统的参数集。

Y 从某个初始值出发，在确定的参数下作自由演化，在 $\mathbf{R}^n$ 中得到一条轨道 $f(y)$，称为期望轨道或者参考轨道。产生 $f(y)$ 轨道的系统 $\dot{y} = f(y,p^*)$ 称为目标系统或者参考系统，现对系统 $\dot{x} = f(x,p)$ 的参数集 P 引入一定量的微扰来修改系统的演化，使得 $f(x)$ 和 $f(y)$ 同步。

对参数 P_j 引入的控制律为：

$$P_j = -\varepsilon \sum h\left[(x_i - y_i), \operatorname{sgn}\left(\frac{\mathrm{d}f_i}{\mathrm{d}p_i}\right)\right] - \delta g(p_j - p_j^*)$$

其中，$i = 1,2,\cdots,L(L \leqslant n)$；$j = 1,2,\cdots,K(K \leqslant m)$。式中 p_j^* 为已知的期望轨道 $f(y)$ 的参数值。函数 h、g 分别是 $(x_i - y_i)$ 和 $(p_j - p_j^*)$ 的连续函数，$\operatorname{sgn}(x)$ 为 x 的符号函数，ε、δ 为大于零的常数。

把控制律和系统组合成一个系统：

$$\begin{cases} \dot{x} = f(x,p) \\ P_j = -\varepsilon \sum h\left[(x_i - y_i), \operatorname{sgn}\left(\dfrac{\mathrm{d}f_i}{\mathrm{d}p_i}\right)\right] - \delta g(p_j - p_j^*) \end{cases}$$

研究表明，只要上式的 Lyapunov 指数都为负值，则状态变量 x 和 y 可以达到同步。因此，可由最大的 Lyapunov 指数为 0 这一条件来确定 ε、δ 两个常数的取值范围。

要使系统的实际输出与所规定的目标输出能更好、更快地达到目标，h、g 的函数形式

的选择是十分重要的。从理论上讲，函数 h、g 有多种选择形式，但在实际中，一般都取：

$$\begin{cases} h\left[(x_i - y_i), \operatorname{sgn}\left(\dfrac{\mathrm{d}f_i}{\mathrm{d}p_i}\right)\right] = (x_i - y_i)\operatorname{sgn}\left(\dfrac{\mathrm{d}f_i}{\mathrm{d}p_i}\right) \\ g(p_j - p_j^*) = p_j - p_j^* \end{cases}$$

2. 主动-被动同步法

主动-被动同步法也是同步控制的一种重要方法。该方法是选择一个由多个变量组成的驱动函数来驱动实现同步的，因此不需要将混沌系统分解，具有一定的灵活性和普适性。

将式（6-3）变形为如下形式，作为系统 1。在 x 的表达式中取

$$s(x) = -ax + (a - b)y - ah(x)$$

作为驱动函数复制系统 2：

$$\text{系统 1：} \begin{cases} \dot{x} = ax + bx + [-ax + (a - b)y - ah(x)] \\ \dot{y} = x - y + z \\ \dot{z} = -\beta y \end{cases}$$

$$\text{系统 2：} \begin{cases} \dot{x}' = ax' + by' + s(x) \\ \dot{y}' = x' - y' + z' \\ \dot{z}' = -\beta y' \end{cases}$$

系统 1 和系统 2 的误差函数为：

$$E = (e_1, e_2, e_3)^{\mathrm{T}} = (x' - x, y' - y, z' - z)^{\mathrm{T}}$$

由此得到：

$$\dot{E} = YE$$

其中，$\boldsymbol{Y} = \begin{pmatrix} a & b & 0 \\ 1 & -1 & 1 \\ 0 & -\beta & 0 \end{pmatrix}$，矩阵 $\boldsymbol{Y}$ 的特征值方程为 $|\boldsymbol{Y} - \lambda \boldsymbol{I}| = 0$，即

$$\lambda^3 - (a - 1)\lambda^2 - (a + b + \beta)\lambda - \beta a = 0$$

由混沌同步的定理可知：当矩阵 $\boldsymbol{Y}$ 的 3 个特征根的实部都小于 0 时，两个系统就能达到同步。用韦达定理可以大概估计 a、b 的取值范围（必要条件）为 $a < 0$，$b < \beta - a$。由于是估算的必要条件，实际取值时应该取得更小些较好。

6.2 混沌密码原理

经过多年的研究，学者们逐渐达成了一种共识：混沌作为一种非线性现象，有许多独特的，值得利用的性质，或许能够为密码学的发展提供新的思路，为保密通信提供更好的手段。目前的研究也发现，传统的密码方法中存在着与混沌的联系；与此同时，混沌现象也具有密码的某些特征。因此，研究混沌保密通信，不仅对构造新的更安全的加密方法和加密体系有帮助，对进一步深入地理解现有的密码与密码体制也有帮助。

一个密码系统其实也是一个映射，只是它是定义在有限域上的映射。密码系统是一个确定性的系统，它所使用的变换由密钥控制。加密变换 $c = e(k,p)$ 的求取不困难，但在不知道 k 的情况下，解密变换 $p = d(k,c)$ 的求取却极为困难。要做到这一点，不但密码系统须对密钥极端敏感，同时密文也须对明文非常敏感，这使得在知道部分密文（和明文）的条

件下，猜测全部明文（或密钥）极其困难。要保证这一点，明文须得到充分的混合。这些对密码系统的要求和混沌的特性有着十分密切的联系。

实际上，一个好的密码系统也可以看成是一个混沌系统或者是伪随机的混沌系统，如表6-1所示。比如典型的DES加密算法，它采用的S加密盒和P加密盒，其实就是一类确定性的类随机置乱操作。

表6-1　混沌理论与密码学的关系

	混沌理论	密码学
相同点	对初始条件和控制参数的极端敏感性	扩散，通过混合打乱明文统计关系
	类似随机的行为和长周期的不稳定轨道	伪随机序列
	混沌映射通过迭代，将初始域扩散到整个相空间	密码算法通过加密轮产生预期的扩散和混乱
	混沌映射的参数	加密算法的密钥
不同点	相空间：实数集	相空间：有限的整数集

下面探讨如何利用混沌设计流密码与分组密码的问题。

1. 利用混沌系统构造流密码

混沌流密码实际上是利用混沌映射产生一个混沌序列，然后使用该混沌序列和明文作某种可逆运算，如异或运算，从而完成加密。如果按照香农所提出的一次一密乱码本的思想，序列密码的密钥长度须长于被加密信息的长度，而这在实际使用中是没有意义的。于是，问题就变成了寻找短的密钥，产生具有足够长周期的伪随机序列的问题。这样构成的混沌序列密码系统，其安全性在很大程度上取决于伪随机序列的随机性。但是，要产生足够复杂，难以用数学工具寻找到规律的伪随机序列其实是很困难的。因为，对于伪随机序列来说，总存在着某种隐性结构，从这个角度来说，一个好的伪随机数发生器就是要具有更好的隐性结构，或者说，具有更难以用统计方法检测出的结构，使得对特定的应用来说，这个伪随机数发生器的内部结构更难以被发现。混沌映射由于其所固有的伪随机性和遍历特性，很自然地成为了伪随机数发生器的候选者。目前，已经提出了许多基于混沌的伪随机数发生器（CPRNG）和混沌伪随机二值序列发生器（CPRBSG）的构造方法，以及基于混沌的流密码。但这些CPRNG或CPRBSG大都是利用离散的混沌映射在连续域上实现的，很少考虑数字实现的问题。而一般所采用的流加密基本上是在有限域上实现的，当连续域上的混沌映射数字化以后，其性能将下降。譬如用计算机生成混沌伪随机数，则原来没有周期的混沌序列将出现周期性的重复，且周期长度是随机的。目前对该周期长度的分析尚没有理论结果，数值模拟表明周期长度和计算精度与初值选取有关。具体来说，假设采用N位长精度来实现混沌映射，所产生的混沌序列的最大周期长度记为M，则有$0 < M(N) < (1/N)^{\varepsilon}, \varepsilon = 0.68 \pm 0.05$。其实，混沌映射从理论上说，是在连续域上的一种映射，它没有固定的周期点（或者说它有从周期为1一直到周期为无穷的所有周期点），且在各个点都呈现不稳定的状态。但是，当这样的映射数字化以后，运动轨迹会在离散的相空间里呈现稳定状态，映射重新出现周期。很多研究者发现了这个问题，并试图解决它，但不幸的是，至今还没有一个好的理论结果。也有一些工程方法被提了出来，比如提高计算精度，将多个混沌系统串联起来，以及基于扰动的算法等。

在传统密码学中，出于硬件可实现性的考虑，以及便于安全性分析，采用的是利用线性

反馈移位寄存器产生 m 序列的方法。该方法简单易行，但不够安全。而实际上，要想产生足够复杂的伪随机序列，使得一般的统计分析方法不能够找到蕴藏其中的规律，同时，该序列的产生又不太复杂，则只有求助于某种非线性系统，尤其是具有混沌特性的非线性系统。因为只有非线性系统才能在看似简单的系统中产生复杂的行为（如 Logistic 映射），而要满足伪随机序列的遍历性，则又需该系统具有混沌特性。

（1）混沌序列的生成

流密码的目的就是要产生一系列随机的密钥值，并且密钥流必须具有随机性，同时它还应在接收端能够同步生成，否则不能实现解密。多数实际的序列密码都围绕 LFSR 进行设计。由线性反馈移位寄存器所产生的序列中，有些类似 m 序列，具有良好的伪随机性，人们开始曾认为它可作为密钥流，但很快又发现它是可预测的，其密码强度低。

混沌系统具有产生密钥流的天然的优良品质：能产生对参数、初始值敏感的混沌值。所以用混沌系统产生密钥流是很好的方案。可以用混沌系统产生随机实数值序列、伪随机二值序列、位序列、四值序列。同时可以由上述各种二进制序列构成随机数序列。所以既可以用混沌系统设计伪随机二进制序列发生器，也可以用混沌系统设计随机数发生器。

（2）混沌实数值序列

任何一个混沌系统在一定的条件下都可以产生混沌实数值序列。可以把直接产生的实数值序列作为密钥流用于加密明文信息，但是对于一般混沌系统而言，其实数值序列分布是不均匀的。下面以 Logistic 系统 $x_{n+1} = \mu x_n(1 - x_n)$ 为例来看看实数值序列的特点。系统中 $u = 3.95$，$x_0 = 0.31415$。统计产生的混沌实数值序列如图 6-5 所示。

通过对 Logistic 映射分布更详细的统计，可以得到 Logistic 映射的分布是很不均匀的。所以，直接将数值序列用作密钥流是不可取的。不仅如此，Logistic 映射的相邻点也有非常强的相关性。

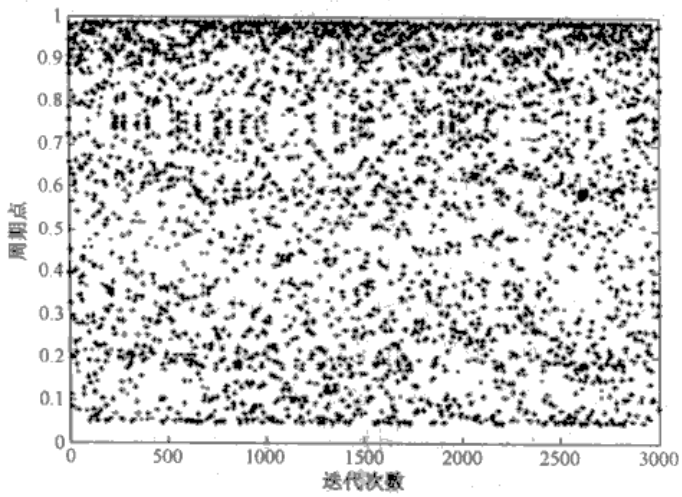


图 6-5　Logistic 序列分布（$u = 3.95$，$x_0 = 0.31415$）

（3）混沌伪随机序列的设计

在实际应用中经常利用混沌系统来产生伪随机二值序列。我们知道，一般混沌系统产生的实数值序列是不均匀的，如果直接用作密钥流，势必会影响加密的效果。但是可以通过一些构造方法对实数值序列进行必要的处理，从而产生伪随机二值序列（PRBS）。其中用得最多的就是相空间分割法，下面进行说明。

假设动态系统 (X,φ) 有一个归一化的不变测度 μ，现在将空间 X 分成不重叠的两个部分 X_0 和 X_1，使得 $\mu(X_0) = \mu(X_1) = 1/2$。以初值 $x_0 \in X$ 为种子在 φ 和 X_0 的约束下开始演化。假设经过 n 轮迭代产生 X_n，则二值序列的第 n 比特 b_n 由下面的公式决定：

$$b_n = \Theta(x_n) = \begin{cases} 0, & x_n \in X_0 \\ 1, & x_n \in X_1 \end{cases}$$

这样得到一个二值序列 $\{b_0,b_1,L,b_n\}$。

根据上述定义产生伪随机二值序列。为了得到性能很好的混沌二值序列，必须找到很好的方案来划分相空间。如果相空间的划分不合理，那么得到的混沌序列就不好。同样以 Logistic 映射为例，由于 Logistic 映射的相空间是（0，1），所以，可以任意选取一个值 $x \in (0,1)$ 来划分相空间。根据不同的取值得到的不同混沌二值序列。如表 6-2 所示。

表 6-2　Logistic 比特序列统计表

x 的取值	0 的个数	1 的个数	0 与 1 个数的比例(%)
0.35	7065	2935	41.54
0.4	6676	3324	49.79
0.5	6048	3952	65.34
0.52	5920	4080	68.92
0.59	5455	4545	83.32
0.6	5276	4724	89.54
0.625	5020	4980	99.20
0.628	4995	5005	99.80
0.7	4317	5683	75.96
0.75	3883	6117	63.48

从表 6-2 可以看出，对相空间的划分是非常关键的。根据二值序列质量评价标准，应该尽量选择能生成 0 和 1 的个数相等的序列的参数。由于划分相空间的参数不同，得到的二值序列不同，因此在加密过程中，可以把划分相空间的参数也作为密钥。

（4）位序列设计

这种方法的思想就是把混沌实数值序列转化为一定长度的浮点数形式而得到：

$$| x_k | = 0.b_1(x_k)b_2(x_k)\cdots b_i(x_k)\cdots b_L(x_k)$$

其中，$b_i(x_k) \in [0,1]$ 是 x_k 的第 i 位，所需的序列即为 $\{b_i(x_k)\}$，$i = 0,1,2,\cdots,L,k \in \boldsymbol{Z}_q^*$。这样混沌系统每迭代一次就可以获得 L 比特长度的二值序列，在获得同等二值序列的情况下，混沌系统的迭代次数仅是移位寄存器生成方式的 $1/L$，大大地减少了获得混沌位序列所需的计算量。

对于每个混沌实数值转化成二值序列还可以作部分改动。对每个实数值不取全部的二进制位，而是引进位抽取函数，对每个实数值只抽取部分二进制位，如只取偶数位或奇数位等，这样可以增大密钥强度。

上述的序列设计是针对混沌实数值在（0，1）区间的情况，那么推广到普遍情况可以按如下方法生成位序列：

假设由一维混沌映射 $x_n = f(x_{n-1}),x_n \in [d,e]$ 获得的序列为 $\{x_n \mid x_n = f^n(x_0) \in [d,e]\}$，对任意 x_n 有 $(x_n - d)/(e - d) \in [0,1]$，表示成二进制为：

$$\frac{x_n - d}{e - d} = 0.b_1(x_n)b_2(x_n)\cdots b_i(x_n)\cdots$$

其中，$b_i(x_n) \in \{0,1\}$。这样得到一个二进制序列 $\{b_0,b_1,\cdots,b_n\}$：

$$b_1(x_n) = \sum_{r=1}^{2^{l-1}} (-1)^{r-1}\Theta_{(e-d)(r/2^l)+d}(x_n)$$

其中，$\Theta_t(x_n) = \begin{cases} 0, & x_n < t \\ 1, & x_n > t \end{cases}$

（5）混沌随机数发生器的设计

随机数发生器是生成序列密码的重要部件，它的好坏直接影响到密钥强度。对一个伪随机序列一般有如下的性能要求：

1）对种子数敏感，即任意两个不同的种子数，产生的序列具有很大的差异。

2）概率分布均匀。

3）数据点之间统计独立，即在已知点 $\{X_i, X_{i+1}, \cdots, X_{i+k-1}\}$ 的条件下预测 X_k 是困难的。

4）序列没有周期。

传统的伪随机发生器（Pseudo-Random Number Generator，PRNG）使用线性同余随机数产生器（Linear Congruential Generator），它可用公式表述如下：

$$x_{n+1} = (ax_n + b) \mod N$$

此处，N 是一个自然数，$x_n, a, b \in \{0, 1, \cdots, N-1\}$。可以证明，线性同余随机数发生器是有周期的，其周期最大为 N，并且，当下面条件之一满足时，可达到最大周期：

1）b 和 N 互素。

2）如果 $N \mid p$，则 $a-1$ 须为 p 的倍数。

3）若 N 为 4 的倍数，则 $a-1$ 须为 4 的倍数。

上述 $x_{n+1} = (ax_n + b) \mod N$ 可以看成是对映射

$$x_{n-1} = (ax_n + b) \mod 1, \ x_n \in \{0, 1\}$$

的数字化，而原始映射在 $a > 1$ 的时候是混沌的。一些传统的伪随机数发生器本身就具有混沌的特性。因此，很自然也可以考虑采用混沌映射来产生伪随机数。

尽管混沌映射具有内在的伪随机性，但是，直接利用它作为伪随机数发生器仍然存在一些问题。比如，它产生的数据序列不一定均匀分布，相邻的数据点之间具有高度的相关性。

2. 利用混沌系统构造分组密码

混沌的特性与密码变换的特性在很多方面是一致的：

1）两者都是确定性的变换，且都会表现出某种类随机特性。

2）混沌映射具有拓扑传递性，而密码变换具有混合特性。

3）混沌对初值和参数敏感，这个特性使得两个初值很接近或者参数有微小变化的混沌映射的轨道会呈指数形式分离，而密码变换则要求对密钥敏感，密钥或明文的微小变化会带来密文的显著不同。

4）混沌是通过多轮的迭代来获得指数分离的轨道，而密码则通过多轮的置乱与混合将明文打乱。

密码系统中的很多特性，在混沌映射中能找到对应点。但是，混沌与密码系统之间仍然存在着许多不同，最大的不同之处就在于：混沌是定义在连续闭集上的，而密码系统的操作只限于有限域。而正是这一点的不同，造成混沌系统不能采用数字化的方法直接应用于加密系统。同时，也不是所有的混沌系统都适合用来设计密码。因此需要巧妙地设计，才能将混沌映射用于密码系统。

由于分组密码的设计就是进行明文和密文的置换，为了达到很好的置换效果必须很好地利用密码学中的两个基本原则：混淆和扩散。由于混沌具有拓扑传递性，因而具有混迭特

性。所谓映射 f 具有混迭特性是指：对任意两个可测集 A_1 和 A_2，有 $\lim_{n\to\infty}\mu(f^{-n}A_1 \cap A_2) = \mu(A_1)\mu(A_2)$ 的关系。也就是说，任何一个非零测度初始集合在映射 f 的作用下，在演化过程中将扩散到整个相空间。这一点和分组密码设计中要求的扩散作用相对应。如果考虑将明文空间作为映射（对应加密变换）的一个初始区域，那么混沌的混迭特性意味着可以将明文的一位扩散到整个密文的每一位上去。

同时，混迭的系统还有一个有用的特性：如果 μ_0 是对 μ 连续的任意一个测度（假设它已经归一化），且 $\mu_n = \mu_0(F^{-n}A)$，则对任意可测集 A 有 $\mu_n(A) \to \mu(A)$。也就是说，在一个具有混迭特性的动态系统中，任何初始非均匀分布的集合都将趋于均匀分布。因此若采用具有混迭特性的系统来加密，当迭代趋于无穷时，由于密文的分布已经不同于初始的明文分布，密文的统计特性不会依赖于明文。这一点正好符合分组密码设计的混淆原则。由此可见，将混沌映射用于分组密码，实际上就是应用混沌的混迭特性来快速地混淆和扩散数据。一般，在设计分组密码时通常采用“置乱”和“替换”方法来混淆和扩散数据，很多分组密码系统都采用这两种方法，如目前广泛应用的 S 盒和 P 置换。

（1）猫映射

猫映射是 Arnold 和 Avez 在 1968 年提出的一个离散混沌映射。该映射的几何解释如图 6-6 所示。

猫映射的数学表达式如下：

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \boldsymbol{A}\begin{pmatrix} x_n \\ y_n \end{pmatrix} (\mathrm{mod}1),\ \boldsymbol{A} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

其中，(mod1) 表示只取实数的小数部分。由 $\det A = 1$ 可知该映射为一个保面积映射，且由式

$$\begin{vmatrix} 1-\sigma & 1 \\ 1 & 2-\sigma \end{vmatrix} = \sigma^2 - 3\sigma + 1 = 0$$

可求得该二维映射的两个 Lyapunov 特性指数分别为 $\sigma_1 = (3+\sqrt{5})/2 > 1$，$\sigma_2 = (3-\sqrt{5})/2 < 1$。由于至少有一个 Lyapunov 指数大于 1，因此该映射是混沌的。一幅图像在猫映射的作用下先进行线性拉伸，然后通过取模运算进行叠加，如此循环往复，最终达到混乱。为了将猫映射用于加密，需要对它进行一下处理。

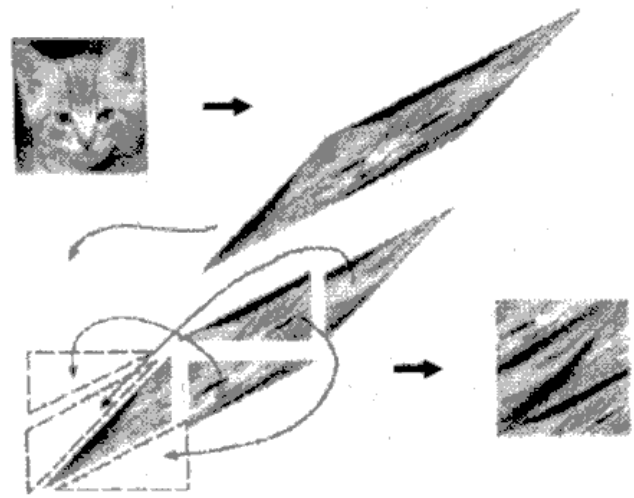


图 6-6　猫映射

首先，将猫映射扩展到 $N \times N$，并进行离散化，如下所示：

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \boldsymbol{A}\begin{pmatrix} x_n \\ y_n \end{pmatrix} (\mathrm{mod}N)$$

其次，引入加密参数。加密参数的引入可以通过改变矩阵 $\boldsymbol{A}$ 的元素来获得，如令

$$\boldsymbol{A} = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$$

为了保证一一映射的特性，有约束条件 $|ad - bc| = 1$。为了工程实现的可行性，可以简

化变换矩阵 A，采用矩阵

$$A = \begin{pmatrix} ab+1 & a \\ b & 1 \end{pmatrix}$$

其中，a、b 都取小于 N 的整数。

（2）混沌分组密码的设计方法

从上文中把猫映射用于加密可以看出：将混沌映射用于分组密码，实际上就是应用混沌的混迭特性来快速地置乱和扩散数据。一般，在设计分组密码时利用“扩散”、“置乱”和“替换”等方法设计单轮加密变换，然后，通过对该加密变换 E 进行多轮迭代，达到扩散和混淆明文的目的。

目前并没有公认的混沌分组密码设计的一般理论和方法，但是有一些方法及准备值得参考：

1）选择一个混沌映射：要求所选混沌映射具有好的混迭特性、大的参数空间以及稳定的结构。

2）引入加密参数：也就是考虑哪些参数可以用作密钥，参数范围是什么，以及如何选择参数保证系统的混沌特性。

3）离散化该混沌映射：将原始的连续映射离散化，在这一过程中，需保证数字混沌保持原混沌的混迭特性，并检查数字化后映射的周期性。

4）密钥的分配：合理地将混沌映射的控制参数与密钥对应起来，保证获得足够大的密钥空间。

5）进行密码分析：一般是利用尽可能多的密码攻击方法对系统的安全性进行测试。但即使通过了所有已知的密码分析方法，仍然不能保证该密码系统是安全的。

上面叙述的法则是一个一般化的方法，在具体设计时，还有很多要考虑的。

值得注意的是，并非所有的混沌系统都适合用来作为加密系统，一般来说，符合下面 3 个条件的混沌系统比较适合用来作加密：

1）具有好的混迭特性。

2）为鲁棒混沌或者至少是结构上稳定的动态系统。

3）具有大的参数集。

其中所谓鲁棒混沌或结构稳定的动态系统是指参数在具有小扰动的情况下获得的混沌映射和原混沌映射具有拓扑等价性。比如，Logistic 映射 $x_{n+1} = \lambda x_n(1 - x_n)$ 就不是一个鲁棒混沌系统。目前还没有一个或一些指标可以用来定量评价混沌系统可用于密码系统的程度。但是可以从 Lyapunov 指数和混沌映射的分布均匀性角度给出一些定量的衡量。在选择用于分组密码的混沌系统时，要求混沌系统具有以下几个特性：

1）混沌映射的 Lyapunov 指数尽可能大。

2）混沌映射具有均匀的概率分布。

3）混沌控制参数要多，且参数空间要大。

4）可逆的一一映射对分组密钥具有优越性。

（3）基于混沌的分组密码算法

加密算法按置换操作、扩散操作、密钥产生机制等几个部分分别进行设计。此外，由于加密对象（图像的像素位置坐标和像素值）以及适应硬件实现时 FPGA 的定点运算等要求，

在算法的各环节还将对混沌映射作相应的离散化处理。

1）三维猫映射及置换操作扩展。

a）猫映射的三维扩展。通过引入几个控制参数 a_x、b_x、a_y、b_y、a_z、b_z，对二维猫映射进行三维扩展操作：

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{pmatrix} = \boldsymbol{A} \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} \bmod 1$$

其中，

$$\boldsymbol{A} = \begin{pmatrix} 1 + a_x a_z b_y & a_z & a_y + a_x a_z + a_x a_y a_z b_y \\ b_z + a_x b_y + a_x a_z b_y b_z & a_z b_z + 1 & a_y a_z + a_x a_y a_z b_y b_z + a_x a_z b_z + a_x a_y b_y + a_x \\ a_x b_x b_y + b_y & b_x & a_x a_y b_x b_y + a_x b_x + a_y b_y + 1 \end{pmatrix}$$

为了简单起见，设 $a_x = b_x = a_y = b_y = a_z = b_z = 1$，可以得到原始二维猫映射的直接扩展：

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{pmatrix} = \boldsymbol{A} \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} \bmod 1$$

其中，$\boldsymbol{A} = \begin{pmatrix} 2 & 1 & 3 \\ 3 & 2 & 5 \\ 2 & 1 & 4 \end{pmatrix}$

通过数值计算，可以发现 $\boldsymbol{A}$ 的 3 个特征值是：$\sigma_1 = 7.1842 > 1$，$\sigma_2 = 0.2430 < 1$ 和 $\sigma_3 = 0.5728 < 1$，分别对此 3 个数字取对数，得到该映射的 Lyapunov 指数。可以看到，其中最大的 Lyapunov 指数是严格大于 0 的，即这个拓展的三维猫映射也是混沌的，并且注意到，最大的一个 Lyapunov 指数比它的二维模式的值要大，所以三维是一种更强意义上的混沌映射，能够具有更好的数据混迭性能。

b）离散化。由于图像加密是一种在有限集上的转换操作，为了使一个混沌映射能够适用于图像加密，进行如下操作：

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{pmatrix} = \boldsymbol{A} \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} \bmod N$$

其中，$\boldsymbol{A}$ 值如 a）中的定义，并且 a_x、b_x、a_y、b_y、a_z、b_z 都取正整数。

很容易发现 $|\boldsymbol{A}| = 1$，这就意味着离散后的三维猫映射也是个一一映射，并且它的混迭特性和对初始条件和参数的敏感性保持不变。

不幸的是，并不是混沌的所有有用特性都会在离散化之后保留下来，比如经几轮混沌映射后，加密后的图像又回到了初始状态，即出现了周期性。对此，人们常常用扩散操作来对其进行补偿，从而使映射变得不可逆。

2）扩散操作。

在加密方案中引入扩散操作有两个方面的原因：一方面，扩散过程会使离散的混沌映射不可逆；另一方面，通过将明文图像每一个比特位的影响扩散到整个密文图像，从而在很大程度上改变明文图像的统计特性。对于一个安全的加密方案，扩散机制是必不可少的，否则

对手可以通过比较一对明文和密文来攻击加密系统，从而发现一些有用的信息。

首先，选择两个数：一个记作 L_i，是（0，1）之间的浮点数，用作初始条件；另外一个记作 S，是一个整型数，用作种子。然后，以 L_i 为初始值，用下面的公式进行迭代运算：

$$x(k+1) = cx(k) \bmod 1$$

如果下一次值取在（0.2，0.8）子区间，那么进入下一步；否则，继续迭代直到获得一个满足要求的在（0.2，0.8）区间的值。然后通过适当的尺度空间对它进行数字化，结果记为 $\varphi(k)$。接着，通过下面的公式，将它与图像中的当前像素和前一个像素的值进行异或操作：

$$C(k) = \varphi(k) \oplus \{[I(k) + \varphi(k)] \bmod N\} \oplus C(k-1)$$

其中，$I(k)$ 是当前操作的像素值，$C(k-1)$ 是前一个像素的加密值，$C(k)$ 是当前像素经异或操作后的加密值，而 N 是颜色等级（对一个 256 级灰度图像，$N=256$）。设初始值 $C(0) = S$。上面的逆变换由下面公式给出：

$$I(k) = \{\varphi(k) \oplus C(k) \oplus C(k-1) + N - \varphi(k)\} \bmod N$$

由于在第 k 步，前面的 $C(k-1)$ 的值是已知的，因而 $C(k)$ 是能被解密的。

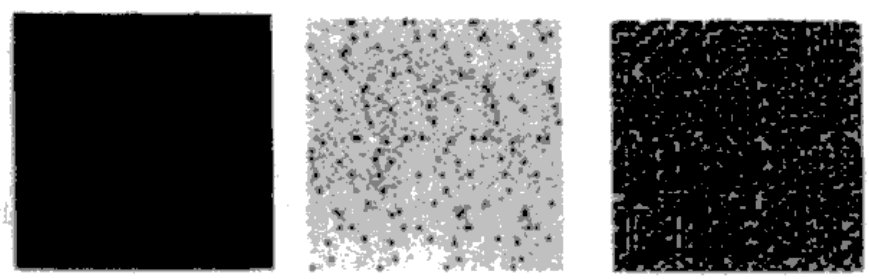
3）密钥机制。

根据密码技术的基本原则，一个密码系统必须对密钥敏感，也就是密文必须与密钥有紧密的联系。有两种方法来实现这一目的：通过加密过程将密钥彻底地混合到密文中去；或是使用一个好的（理想情况下是真正随机的）密钥产生机制。这里采用如下的混沌映射来产生密钥：

$$x(k+1) = cx(k) \bmod 256$$

4）置乱与扩散效果演示。

下面以一幅简单的图像分别演示置乱与扩散的加密效果，如图 6-7 所示，其中置乱操作是对图像的像素位置进行的，而扩散操作是对图像像素值进行的。



a) b) c)

图 6-7 一幅简单图像的置乱与扩散效果

a）原始图像 b）置乱效果 c）扩散效果

6.3 混沌密码的应用

本节将给出混沌密码的两个应用。

6.3.1 基于混沌分组密码的图像加密算法方案

下面给出基于三维猫映射的图像加密方案。完整的图像加密方案由5个操作步骤组成，如图6-8所示。

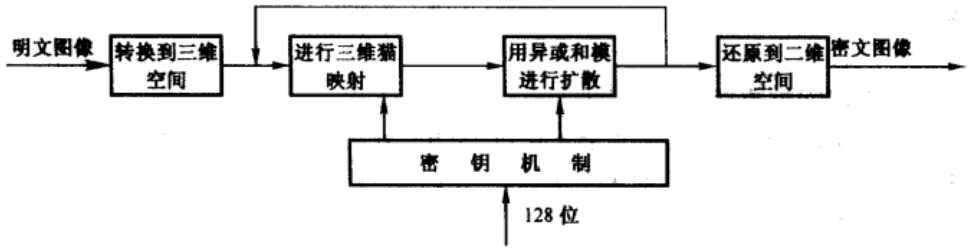


图6-8 三维猫映射的图像加密方案

1）密钥产生。选择一个长为128位的序列作为密钥，将它分成8个组，这些将最终对应到各个混沌映射的几个参数（a_x，b_x，a_y，b_y，a_z，b_z，L_i，S）。

2）将二维图像变换到三维空间。假设被加密的是一个宽为W像素，高为H像素的图像。首先需要将图像的所有像素堆积起来，形成尺寸分别为$N_1 \times N_1 \times N_1$，$N_2 \times N_2 \times N_2$，$N_i \times N_i \times N_i$的几块，同时还必须满足下面的条件：

$$W \times H = N_1^3 + N_2^3 + \cdots + N_i^3 + R$$

其中，$N_i \in \{2,3,\cdots,N\}$是每个立方体的棱长，N是最大允许块的棱长，而$R \in \{0,1,2,\cdots,7\}$是余数。

3）进行三维猫映射。用a_x、b_x、a_y、b_y、a_z、b_z作为三维离散猫映射的控制参数来对每个图像块中的像素坐标进行操作，从而产生置乱后的图像。

4）扩散操作。设$x(0) = L_i, c(0) = S$，然后使用上面提到的算法进行扩散。

5）将三维立方体转换回原来的二维图像。适当地组合三维立方体，将其还原为二维图像，以便于显示和存储。

通常，根据安全性的要求，第3）步和第4）步需要被交替地执行几次。执行的轮数越多，相应的安全性就越高，但代价是计算量的增加和时间上的延迟。解密过程和加密过程类似，使用上述第3）步和第4）步相反的顺序。由于加密和解密的过程相似，它们也具有相同的计算复杂度和时间消耗。

6.3.2 基于混沌的语音加密算法

本节将利用一个二维可逆混沌和一个一维混沌系统，构造一种快速的分组加密算法。该算法利用二维混沌映射对数据块中各元素的位置进行置乱，利用一维混沌映射构造替换表，通过对各元素值的替换达到改变数据块统计特性的目的。

（1）第一步：置乱操作

在加密算法中，首先将一维的信号转化为二维矩阵，然后利用混沌映射的混迭特性对二维矩阵的位置进行置乱。为达到对位置充分置乱的目的，对所选择的混沌变换提出一定的要求：

1）该变换是一个二维的变换。

2）该变换必须是保面积变换，或者说该变换必须是个一一变换。

3）该混沌变换在离散化后仍然能够保证满足遍历性和混迭性。

（2）第二步：替代操作

采用 Logistic 映射构造替代表，然后利用该替代表对置乱后的明文按采样点进行替换。考虑 Logistic 映射：

$$x_{n+1} = 4x_n(1 - x_n)$$

可以按照下面的步骤构造替换表：

1）将区间 [0,1] 均匀分为 256 个子空间 $\{s(0), s(1), s(2), \cdots, s(255)\}$，每个区间分别赋一个整数值 0，1，…，255，即 $V(s(i)) = i$，记 $S(i)$ 的上、下界分别为 $s_d(i)$ 和 $s_u(i)$，$s(i)$ 中的任意数据 s 有 $s_d(i) \leqslant s \leqslant s_u(i)$。

2）任意取一个值为初始值，经过 N 轮初始迭代后，根据数值迭代后所在的区间记录下整数序列。若第 $n(n > N)$ 次迭代获得的数据在区间 $s(k)$ 中，则这次得到的整数值为 k，如此迭代下去，获得一个有 256 个不重复数据的序列 $\{k_0, k_1, k_2, \cdots, k_{255}\}$，且 k_i 属于 $\{0,1,2,\cdots,255\}$。这里需要处理一下迭代中迭代结果落入以前已经走过的区间的问题。对于这种情况，不记录区间对应的整数值，继续迭代下去，直到迭代结果落入另外一个从未走过的新区间为止。

3）如果原来相空间分割所对应的整数序列为 $\{0,1,2,\cdots,255\}$，现在的序列为 $\{k_0, k_1, k_2, \cdots, k_{255}\}$，则找到的映射关系为 $k_i = f(i)$，即为一个替换表。值得注意的是，在区间 [0,1] 中，Logistic 映射有一个不动点 0，当数值落入 0，0.5 或 1 时，就会在下面的迭代中进入这个吸引点中，所以，初始值不能为这 3 个数。另外，当数值迭代到 0 时，需要给一个小的固定偏移，使之离开这个吸引点。

（3）第三步：扩散操作

扩散过程需要利用上面得到的替换表，标记替换表为 $j = f(k)$，可得到下面的加密混迭公式：

$$C_i = f(P_i) \oplus C_{i-1}$$

令 C_0 为初始值，取密钥 k 为初值，即 $C_0 = K$。

解密时用的反混沌公式为：

$$P_i = f^{-1}(C_i \oplus C_{i-1})$$

令 C_0 为初始值，取密钥 k 为初值，即 $C_0 = K$。

注意到在各种加密模式中，分组大小一般是 64bit 为一组，因此，每 8 字节设定 $C_i = C_0(i = 9,17,25,33,\cdots)$，同时在每次循环过程中都改变 C_0 值。这样在加密时，明文改变一位，密文将以 8 为底数，循环次数为指数进行扩散。解密时明文也对初始的 C_0 值比较敏感。

（4）第四步：加密算法

下面以语音加密为例，完整地阐述加密过程。

1）设置加密密钥：猫映射的映射参数 a、b 和迭代轮数 k；Logistic 映射的初始值；替代操作的初始值 K。

2）构造一个由 Logistic 映射产生的替换表（替换表长度由每个数据点的表示位数决定，如对 PCM 数据，表长为 256bit）。

3）将一个长度为 m 的数据块分为 $m = NN$，比如一段音频数据块的长度为 $m = 1600$bit，

则 $N=40$。

4）利用离散化的猫映射进行位置置乱。

5）在每轮置乱之后，进行一次替换和混迭操作。利用步骤 2）得到的替换表，采用下列混迭公式进行替换和混迭操作：

$$C_i = f(P_i) \oplus C_{i-1}$$

其中，C_0 为初始值，取 $C_0 = K$。若 $i \bmod 8 = 0$，则 $C_i = C_0$。

6）跳转到第 4）步，然后第 4）、5）步循环执行 k 轮。

7）将分块数据重新堆叠成一个列向量，输出。

（5）第五步：解密算法

解密过程与加密过程相似，只是反变换公式有区别：

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = A^{-1} \begin{pmatrix} x_n \\ y_n \end{pmatrix} (\bmod N)$$

其中，$A^{-1} = \begin{pmatrix} 1 & -a \\ -b & ab+1 \end{pmatrix}$

6.4 小结

本章首先描述了混沌学的基本原理，这也是基于混沌运动的各种系统的基本原理。在此基础上，给出了目前在各种混沌系统中经常用到的，也是非常重要的几种混沌控制和同步方法。这些方法是在设计基于混沌的密码系统的时候需要考虑和选取的。然后，介绍了如何将混沌的各种理论应用到密码学的实践中，并对基于混沌的流密码和分组密码的设计方法和需要注意的问题进行了深入的探讨。混沌系统产生的序列从严格意义上讲属于流密码，但和传统的流密码又有区别。因为混沌系统具有的良好的伪随机性、轨道不可预测性等特性，使得由它构造的密码体制具有相当广泛的应用前景。

6.5 习题

1. 什么是单混沌加密方案，分析单混沌加密方案的缺点以及对这种方案的攻击方法。
2. 阐述混沌密码与一般流密码的区别。
3. 判定 $F(x) = 8x^4 + 4x^2 + 1$ 在 $[-1,1]$ 上是否是混沌的。
4. 证明映射 $B(x) = \begin{cases} 2x-1, & 1/2 \leqslant x \leqslant 1 \\ 2x, & 0 \leqslant x \leqslant 1/2 \end{cases}$ 在 $[0,1]$ 上是混沌的。
5. 设 μ 的取值区间为 $[0,5]$，用 MATLAB 画出 Logistic 迭代的极限形态图。
6. 对于迭代模型

$$\begin{cases} x_{k+1} = 1 + y_k - 1.4x_k^2 \\ y_{k+1} = 0.3x_k \end{cases}$$

取初值 $x_0 = 0$，$y_0 = 0$，进行 4000 次迭代，对于 $k > 1000$，在 (x_k, y_k) 上亮一点，用 MATLAB 绘制 Henon 引力线图。

第 7 章　DNA 密码技术

近年来，DNA 所具有的超大规模并行性、超高容量的存储密度以及超低的能量消耗引起了人们越来越浓厚的兴趣。将 DNA 的性质应用于分子计算、数据储存以及密码学等领域成为了新的研究方向。在这些方面的研究有可能最终导致新型计算机、新型数据存储器和新型密码系统的诞生，从而引发一场新的信息革命。DNA 密码就是在这种背景下随着对 DNA 计算（也称为分子计算或生物计算）的研究而诞生的。传统密码技术在 20 世纪伴随着电子技术的快速进步得到了巨大的发展，也是目前广泛应用的密码系统；量子密码技术诞生于 20 世纪 70 年代，近年来有了一定的发展，但是距离实际应用尚有距离；DNA 密码技术是在 1994 年 Adleman 提出 DNA 计算之后才开始得到世人的关注。由于 DNA 密码的特性，使得它成为了国际密码学研究的前沿领域。DNA 密码、传统的密码和量子密码技术以各不相同的方式，实现着共同的目标——信息安全。它们有可能成为未来密码学的三大主要领域。本章将扼要地阐述生物分子计算，并详细地介绍 DNA 密码学的理论基础及 DNA 在密码学上的应用。

7.1　概述

7.1.1　生物分子计算

DNA 计算是近年来兴起的新的研究领域，其核心就是利用重组 DNA 技术来进行计算。一般来说，生物分子计算（BMC）就是利用诸如 DNA 重组等生物技术所进行的计算。它广泛应用于 DNA 和 RNA 链的处理中，包括定点编辑和剪接操作。BMC 方法的应用之一就是解决复杂的查找问题，如哈密尔顿路径问题。它是通过重组 DNA，在大量可能的解决方案（以 DNA 链的形式表示）中，以大规模的并行方式进行组合搜索来得到问题的解。例如，Boneh 和 Adleman 等人提出了破译数据加密标准（DES）的 BMC 方法。尽管这些解决基于组合查找的难题的办法对于固定大小的问题来说是有效的，但其空间需求最终限制了这些方法的应用范围，因为空间需求会随着输入规模按指数规律增长。尽管如此，除了纯粹的组合搜索，BMC 还有许多应用。比如，由于很少的一点介质就可以携带大量的数据，这使得 DNA 和 RNA 成为非常有吸引力的数据存储载体。它们的存储能力远远超过普通的电子、磁性和光纤存储材料。1 克 DNA 就包含 1021 个碱基，或者说 108TB 的数据存储能力。所以几克 DNA 就能够存储世界上现有的所有数据。大多数重组 DNA 技术的应用环境是 5‰的 DNA 浓缩物水溶液。DNA 链具有以下功能：①可以用 DNA 链构成液态的生物数据库；②可以用 DNA 链构成用以表达二进制数据的生物数据。

对于前者①，因为生物的自然 DNA 可以使用非标准基重新进行编码，因此这样处理的 DNA 可以用于生物分子计算。而对于后者②来说，通过许多不同的方法（如 DNA 芯片阵列等），DNA 数据可以存储在普通的二进制存储介质中；同时，利用寡核苷酸的码表，二进制

数据能够以 DNA 链的形式进行编码。Baum 基于杂交的思想讨论了在 DNA 数据库中的快速相关搜索方法。其他 BMC 技术还可对 DNA 数据进行更为复杂的数据库操作，例如数据库的联合操作和对 DNA 数据的各种大型的并行操作。

7.1.2 DNA 密码、传统密码和量子密码的比较

1. 发展状况

传统密码可以追溯到 2000 多年前的凯撒密码甚至更早，已经基本建立了比较完善的理论体系，目前实际使用的密码都可以算是传统的密码。量子密码诞生于 20 世纪 70 年代，已经有相当的理论基础，但在实现上困难较多，基本还没有投入实际应用。DNA 密码只有不到 10 年的历史，理论尚处于探索阶段，使用代价也比较高昂。

2. 安全性

传统的密码除了一次一密以外，都只具有计算安全性。也就是说，如果攻击者有无限的计算能力，理论上就可以破译这些密码系统。研究表明，量子计算机具有惊人的计算潜力。虽然目前还不能完全确定量子计算机的计算能力，但是存在这样的可能性，即在未来的量子计算机的攻击下，传统的密码中只有一次一密仍然是安全的。量子密码是在现有的理论上不可破译的密码，它的安全性建立在海森堡测不准定理之上，物理法则保证了这个量子信道的安全性。即使窃听者能够做他想做的任何事情，并且有无限的计算资源，甚至 P = NP，他都不能破译量子密码。任何对量子密码的窃听都会造成密码的改变从而被发现；攻击者无法复制出一个和他所截获的量子完全一样的量子，所以要想不被发现地篡改也是不可能的。因此，通过量子密码进行密钥协商，具有无条件的安全性。DNA 密码主要是以生物学技术的局限性为安全依据，与计算能力无关，因此对量子计算机的攻击也是免疫的。但是这种安全性有多高，能够保持多久，还有待于研究。

3. 使用功能

传统的密码使用最为方便，计算过程中可以使用电子计算机、DNA 计算机甚至量子计算机；在传输过程中可以使用电线、光纤、无线信道甚至信使；储存媒介可以使用光盘、磁介质及 DNA 等任何可以存储数据的媒介，并且可以实现公钥加密、私钥加密、身份认证和数字签名等诸多功能。量子密码是在量子信道上实现的，适用于实时通信，不适用于安全数据储存，难以实现像传统密码那样可以轻松实现的公钥加密和数字签名等功能。以目前的技术，DNA 密码只能用物理的方法传送。但 DNA 所具有的超大规模并行计算能力、超低的能量消耗和超高密度的信息存储能力，使得 DNA 密码在对实时性要求不高的大规模并行数据加密、安全数据存储、身份认证、数字签名和信息隐藏等密码学应用中具有独特的优势。DNA 也可以用来制作难以伪造的商业合同、现金支票以及身份识别卡等。

由于传统密码、DNA 密码和量子密码都还在发展之中，尤其是后两者都还有很多问题没有研究清楚，目前很难准确预测未来密码界的发展。但从上述分析可以看出，在未来相当长的时间内，这 3 种密码很可能是互相补充，共同发展，而不是某一个被彻底淘汰。

7.2 DNA 密码学的理论基础

假设明文消息数据以 DNA 链的形式进行加密。例如，可以用试管溶液中的 DNA 链构成

一种液态、加密的生物数据库。本节中将详细论述与 DNA 数据加密技术紧密相关的粘贴模型以及它的物理实现。

7.2.1 粘贴模型的概念

1. 信息的表示

粘贴模型使用两组单链 DNA 分子代表一个比特（bit）序列。对于由 N 个碱基构成的存储序列，可将其细分出 K 个无重叠的区域，每个区域由 M 个碱基构成（因此 $N \geqslant MK$）。在计算过程中，每个区域视为一个比特的位置（或者等价于一个布尔变量）。在模型中有 K 个长度为 M 个碱基的不同的粘贴序列或简单粘贴链，并且这些粘贴链与且仅与 K 个存储区域中的一个互补。如果一个粘贴链退火粘贴在一个给定存储序列的匹配位置上，则该比特串的相应区域就被置位。如果没有粘贴链退火粘贴在某区域上，则该区域的比特就被复位。图 7-1 说明了这种表示方法。

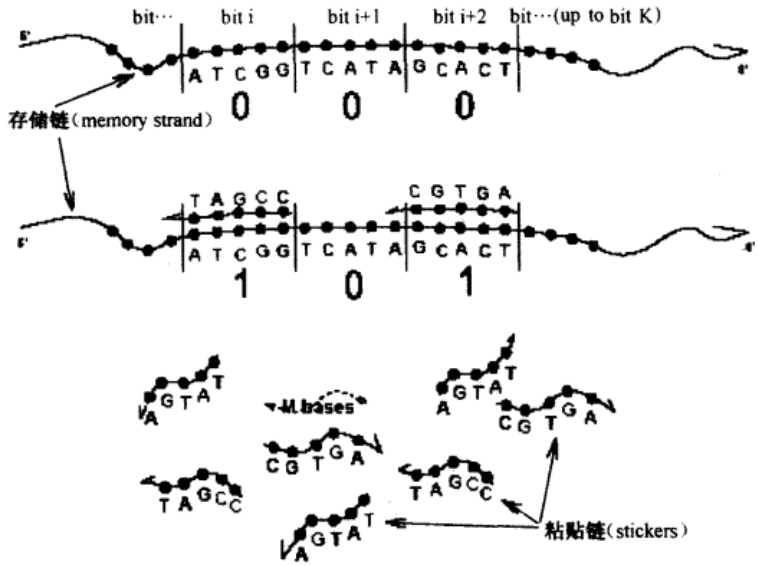


图 7-1 粘贴模型

每个存储序列和它的退火粘贴链（如果有）代表一个比特串。这种局部双链称为存储双链。许多同样的存储序列代表一大批比特串，这些序列只在需要的比特位置上有退火粘贴链。这种存储复合物的一个集合称为管道。这与早先使用 DNA 表示信息的方法有所不同，在那些方法中，用在一个链中是否存在一个特定的子序列来代表相应的一个比特是置位还是复位。在这个新的模型中，每一个可能的比特串由一个惟一的存储序列和粘贴链联合表示，而早先每个比特串由一个惟一的分子表示。

对于像 Adleman 所讨论的对 DES 算法的攻击，如果采用基于粘贴模型的 DNA 计算方法，可能需要大约长 12000 个碱基（N）的存储序列，该序列中有 580 个二进制变量（K），每个变量由 20 个碱基构成（M）。同早先由 Boneh 等人提出的方案相比，这种方案中的存储密度为（$1/M$）比特/个碱基。可以注意到，DNA 中信息存储量的理论最大值为 2 比特/个碱基，但是在一个基于分离的分子计算机中，使用这么高的值需要具有用单个基失配来可靠

分离链的能力。因此作为替代，选择牺牲信息密度以减小试验的难度。

2. 串集合的操作

现在介绍对比特串集合的几种可能的操作，它们的组合可以灵活地实现普通的 DNA 算法。在这里介绍 4 种操作：将两组串集合合为一组新的串集合；将一组串分离为新的两组串；置位或清除一组中每个串的第 k 个比特。根据以上介绍的 DNA 表示法，下面对每种逻辑操给出相应的解释，并通过图 7-2 对这 4 种逻辑给出了更为直观的描述。

1）最基本的操作是将两组比特串集合合为一组。在 DNA 中，这相当于由以前的两个旧的管道合成一个新的管道（包括来自两个输入管道的所有存储复合物，而它们的退火粘贴链则不受影响）。

2）将一个串的集合分成两个新的串的集合。其中一个集合包括所有的某个特定比特被置位的原始串，另一个集合包含了所有的特定比特被复位的串。这相当于分离出该组管道中给定比特区域中有退火粘贴链的复合物，同时，原始输入管道被破坏。

3）将一个集合中的所有串的某个比特置位。粘贴链被退火粘贴到每个复合物的适当的区域（如果已经退火，粘贴则保持不变）。

4）最后，清除一个集合中的所有串的某个比特位。在该组的管道中，该比特的粘贴链将从每一个存储复合物中清除。

这个模型的计算由一系列组合、分离和比特置位/清除操作组成。这一系列操作必须始于某个初始的比特串集合，最后得到一个（可能没有）串的集合，得到的串的集合叫做“答案”。在操作中，包含初始比特串集合的管道称为母管道。为了实现对于粘贴模型计算的理论描述，必须描述怎样建立存储复合物的母管道和怎样从最终管道（可能为空）的“答案”中读出至少一个比特串（或者识别出该管道不包含序列）。为此应该首先考虑如何建立母管道：必须建立一个与串的（K，L）库集合相对应的母管道。这个（K，L）库集合是由所有可能的长度为 L 的比特串补上 $K-L$ 个 0 构成的串的集合，因此，该集合总共有 2^L 个长度为 K 的串。

下面给出一个计算实例，目标是在长度为 L 的串中找寻符合要求的串。对于这样一个问题，可以通过并行处理所有可能的 2^L 个输入和排除其中不符合查找标准的输入来寻找“答案”串。需要注意的是，所设计的存储序列的长度大于 L。前 L 个比特代表输入编码，也是初始库的随机部分，而后 $K-L$ 个比特用来放置中间过程的数据的“答案”串。所有的复合物最初都处于复位状态。在之后的计算过程中，所有的比特都可以读和写。用这种方法，建立一个与生成所有可能输入的相对应的 $K-L$ 库集合母管道。

为了从最终的“答案”集合中读出一个串，必须从答案管道中分离出一个具有固定退火粘贴链（如果有）的存储复合物。否则就认为“答案”管道不包括任何序列。

3. 实例

假设有 B 个袋子，每个袋子中都装有一些物体。设这些物体有 A 种。从这些袋子中找到一个最小子集，每种物体至少包含一个。这个问题的正式表述如下：给定一系列 $\{1, \cdots, A\}$ 的子集 $C = \{C_1, \cdots, C_B\}$，求 $\{1, \cdots, B\}$ 的最小子集 I 使之满足 $U_{i \in I} C_i = \{1, \cdots, A\}$。

用以上定义的模型可以直接解决此问题。建立代表所有可能的 2^B 个袋子的存储复合物，并标记所有包含袋子 i 的子集，其中袋子 i 为包含出现在子集 C_i 中的所有类型物体的袋子。

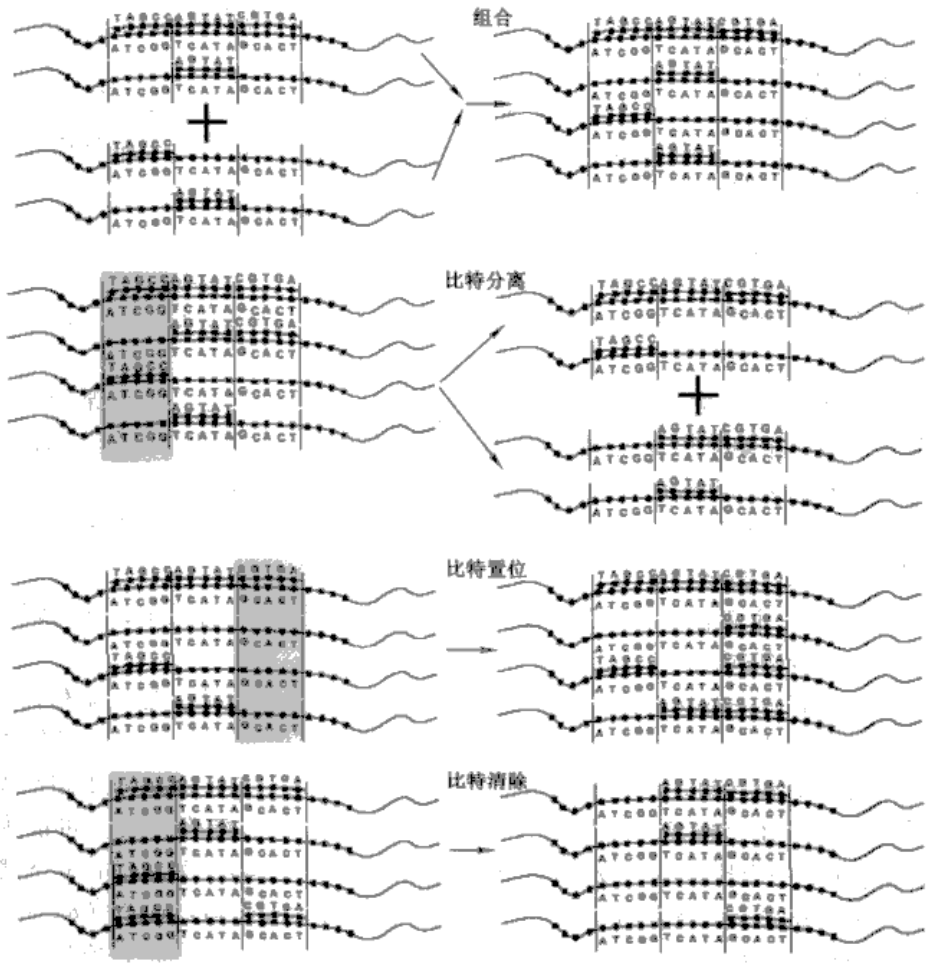


图 7-2　DNA 链的 4 种操作

然后分离出一些被标记为包含所有 *A* 种物体的复合物，并读出其中个数最少的一个或几个。解决这个最小集合覆盖问题的正式的粘贴算法为：

- Design a memory strand with K = B + A bit regions.
- Initialize a (K, B) library set in a tube called T_0.
- for i = 1 to B
  - Separate T_0 into T_{on} and T_{off} based on bit i
  - for j = 1 to | C_i |
    - Set bit N + C_i [j] in T_{on}
    - Combine T_{on} and T_{off} into T_0
- for i = B + 1 to B + A
  - Separate T_0 into T_0 and T_{bad} based on bit i
  - Discard T_{bad}
- for i = 0 to B − 1
  - for j = i down to 0

比特 1…B 表示选择的袋子，比特 (B + 1) … (B + A) 表示物体种类

表示每一个聚合体中的最终 A 的位置…

将不包含所有 A 种类的袋子集合剔除

Separate T_j into $T_{(j+1)'}$ and T_j based on bit i + 1

Combine T_{j+1} and $T_{(j+1)'}$ into T_{j+1}

Read T_1;　　计算用了多少个袋子

else if it was empty then Read T_2;

else if it was empty then Read T_3;

...

其中，$|C_i|$ 是子集 C_i 中物体的数量，$C_i[j]$ 是子集 C_i 中的第 j 个物体。注意，以上算法需要 $O(AB)$ 步，输入为 $O(AB)$ 比特。

可以假设一个自动执行实验的机器人系统，为了控制分子操作，允许此机器人系统可以执行以控制分子操作为目的的任意的连续算法。但是，这些操作必须在“不可见”的情况下执行，也就是说，这些操作与分子操作的接口仅是初始化、联合、分离、置位、清除和读等，除此之外，实验过程中机器人系统不能从 DNA 中得到任何反馈。可以注意到，在通常的理论假设下，粘贴链区域的数量是极大的，粘贴模型具有模拟（并行地）独立的通用机器的能力。粘贴模型是通用的，即使没有清除操作也可以正常地执行，而使用清除却可以使算法更加紧凑。

7.2.2 粘贴模型的物理实现

在我们的模型中，模型的每一步都与 DNA 的存储链及相关的粘贴物之间的操作相对应。下面描述各种操作的物理过程。对于一个给定的操作常常有几种可能的实现方式，每种方式都有优点和缺点。但是，最终将通过实验确定哪种实现方式是可行的。

1. 组合

对于两个管道的组合，可以先对管道内容“再水合（rehydrate）”，然后把液体混合在一起（例如注入或使用水泵）来形成新的管道。应该注意的是，即使是这种看起来非常简单的操作也会受到某些方面的限制：如果这一步操作不够谨慎，注入和混合时的剪切力将会把 DNA 分为大约 15000 个碱基长度的段，从而使得组合操作没有任何意义。

实际上这个操作和其他所有的操作所关注的是由于粘贴在管道、水泵、滴管壁上而“失去”的 DNA 的数量。即使“失去”的 DNA 只是总量的一小部分（对分子生物学家这些并不重要），也会给计算带来影响。

2. 分离

分离操作的最终目标是在物理上分离那些管道中的复合物，使在某个位置上有退火粘贴链的和那些没有任何退火粘贴链的复合物分开。DNA 的配对是实现这一操作的核心机制。一般来说，通过杂交来实现分离是将包含原始存储复合物集合的溶液同许多单独的含有同样复合物的多股探测器链接触。在我们的模型中，每个比特位置有一个特定种类的探测器链（带有一个独特的核苷酸序列），当对该比特执行分离操作时就使用与之对应的探测器链。探测器链序列被设计成仅与其对应的比特的存储序列区域配对而不与其他区域配对。分离过程中，具有特定比特序列的原始复合物将被探测器链俘获，而其他的复合物因为该比特的区域被粘贴链覆盖而仍然游离在溶液中。然后，可以用物理的方法将这些游离（“置位”）的复合物分离出来，例如使探测器链与磁珠结合，或者使探测链附着于载体上，然后进行清洗。最后，“复位”的存储复合物通过洗提从束缚它们的探测器链中恢复出来（通过加热和

冲洗)。以上操作的结果是使母管道分离成为两个新的管道，每个管道中都有一个由分离操作所产生的存储复合物的集合。

注意，如果使用加热来完成最后的洗提过程，那么必须在没有从存储链中除去所有粘贴链的情况下进行。之所以这样，是因为同粘贴链相比，探测器链同其对应区域的亲和力较低。为达到这一目标，可以使探测器链次序同它们在存储器上对应的区域非精确互补，造成探测器链和粘贴链分裂温度的不同。另一个选择是使用次序完全互补的探测器链和粘贴链，但是用另一种主干材料构成粘贴链（例如 PNA 或者 DNG)。同 DNA 探测链相比，它们的结合特性较 DNA 存储序列更为特殊。PNA 和 DNG 可以减少盐浓度，这将使 PNA/DNA 和 DNG/DNA 结合得更牢固，而 DNA/DNA 则相反。因此最终的洗提步骤可以通过在不含盐的溶液中清洗而不是通过加热来实现。当然还有其他方法可以使得粘贴链和探测链具有不同的亲和力。

3. 置位和清除

为了对一个集合中每个串的某一位置位，最显而易见的方法就是对其直接退火。可以将过量的对应粘贴链加入到包含该集合的存储复合物的管道中。粘贴链退火后，粘贴到每个没有对应粘贴链的复合物上。随后，去除过量（未使用）的粘贴链。这一步可以通过过滤或者分离出所有的存储复合物来实现。为了完成以上操作，可以使每个存储序列有一个通用的区域（在起始或者结束的位置)，该区域永远不会被粘贴链覆盖，像上面分离操作中描述的那样，为该区域设计一个探测链。从一个可能包括其他种类链的溶液中恢复所有的存储复合物时，这种设置通用区域的方法一般会比较有效。

在一个管道中，为了清除一个集合中的每个串的某一比特，需要在每个复合物上清除该比特的粘贴链。简单的加热显然无法满足要求，因为所有的粘贴链都会从所有的比特区域脱离。指定某些比特区域为弱区域是一个可行的方法。这些区域的粘贴链是弱粘贴链，同正常的粘贴链相比容易从存储链上分离。当加热到某个中间温度，所有的弱粘贴链就可以分离出来，同时保持正常粘贴链还在适当的位置上。

为了使清除操作更具有一般性，可以通过三螺旋体，利用 PNA 链入侵现象来实现该操作。在适当的条件下，两个全嘧啶 PNA 单绞低聚合物将“侵入”一个存在的 DNA/DNA 互补双链，形成一个（PNA）2/DNA 三螺旋以取代嘧啶 DNA 链。这个过程在有 PNA“夹子”时最有效。该“夹子”在一个分子中同时包含 Watson - Crick 和 Hoogsteen PNA 链。研究表明，如果使用21 核苷 DNA 粘贴链，就要设计长度为 14 个碱基数的 PNA 夹子，它和 DNA 粘贴链的中间 7 个核苷构成一个三螺旋。通过向一个存储复合物管道中混合特定比特对应的 PNA 夹子，然后加热，PNA 夹子将和靶粘贴链形成三螺旋，并使其松动。因此，与未受影响的粘贴链相比，在较低的温度下就可以“撬开”它。但是，这个操作的专一性和可靠性在实验上还是未知的。实际上，三重结构机制可能与把非靶粘贴链保留在适当位置的要求相矛盾。

4. 初始化和最终输出

为了使一个组合库中大致包含每一个可能的长度为 L 比特加 $K-L$ 个 0 的串的拷贝，首先需要粗略合成一个具有 $K \geqslant L$ 个区域的存储链的 2^L 个同样的拷贝。然后粘贴链被“随机”地添加到这些链的第 $1 \sim L$ 个位置上，进而通过以下方法可以一步完成以上所述的过程。

将这些链分成体积相等的两部分。一部分中的 $1 \sim L$ 个比特位被添加过量的粘贴链，结

果使得所有链上的所有比特都被置位。然后清除掉未使用的粘贴链。接下来把这两部分重新合在一起加热，使所有的粘贴链分离。最后将混合物冷却，使粘贴链随机退火粘贴到存储链上。因为每两个链的一个比特的位置只有一个粘贴链，最后得到的存储复合物的每一比特有一半概率被置位（几乎是独立的）。在这个模型中，不在最终库里的某个特定比特串关于 L 的概率是 $(1-1/2^L)^{2L}$，几乎为 $1/e$。换句话说，每个串被生成至少一次的概率大约是63%。显然，初始时可以通过生成多于 2^L 个链来增大这个百分比。

值得注意的是，在化学计量学中，这个过程对误差是相对健壮的。例如，如果原始的链被分成两部分，但是比率不是 1 而是 1.5，那么一个随机选择的链被生成的概率是 37% （$L=56$），仍然不趋近于0。

为了得到一个输出链，就需检测存储复合物在溶液中是否存在。如果存在，就需要分离出至少一个存储复合物，然后确定哪些粘贴链（如果有）退火后粘贴在它上边。

检测复合物可以通过对每个存储链进行荧光标记来完成。然后使溶液通过一个毛细管道，并对单个分子进行检测。这种检测在实验室中已经实现。如果检测事件间隔的时间足够大，这种技术也可以有效地分离单个复合物。除了上边提到的毛细管方法外，其他检测复合物的方法（例如基于 PCR 的方法）也是可行的。

通过直接成像，也有可能确定退火粘贴链，如果知道比特区域的顺序，那么就可以借助一些设备来看和读出“答案串”。一旦一个复合物被分离出来，就可以洗提其粘贴链。虽然这些方法听上去很不错，但是更实际的方法是由 Adleman 提出的基于 PCR 的方法。

5. 存储链和粘贴链设计

在以上的许多讨论中，都涉及了需要设计存储链和粘贴链的顺序使其具有某些特性的问题。这里将归纳、概括这些要求，并探讨实现的可能性。

顺序设计的最基本要求是使粘贴链具有专一性。粘贴链在退火后要严格地粘贴在存储链的相应的位置上。因此在设计存储链时，一定要使得所有区域的互补粘贴链仅仅与该区域互补，而与链上的其他队列只有很少的亲和力。换句话说，在所有其他的队列上需要最小数量的失配。在一条链上，它必须同其他长 M 的窗口失配（可能跨越两个比特区域）。数学上，希望设计一个长 N 的序列，以便存在 K 个长度为 M 的不重叠子序列（称为“区域”），它们具有以下特性：对每个区域来说，在整个序列上，它的互补物同其他每个长度为 M 的子序列有至少 D_1 个失配点。对于一个长为 M 个碱基的粘贴链，要使其不退火，则至少要保证有 D_1 个基失配。

另一个重要的问题是，存储链自身可能形成的次级结构。必须阻止存储链自己退火粘贴形成一个类似于发夹的结构。如果可以在设计上满足这个要求，那么就可以组合设计一个长度为 N 的序列，每个长度为 M 的子序列的互补部分同其他长度为 M 的子序列有至少 D_2 个失配点，这里 D_2 是为了阻止存储链和自己退火粘贴的最小失配数。

最后，必须设计具有足够低的亲和力（相比于粘贴链）的分离探测链，以便它们能粘贴到适当的区域。以此保证每个探测链分离时存在一个清洗温度（和盐度），使得在此温度下，粘贴链可以保留在原来位置。类似的，要求探测链同它们自己对应的区域有 D_3 个失配点，同其他地方有 $D_4 > D_3$ 个失配点。

这些标准看起来比较麻烦。但是，存在一些方法可以比较轻松地达到上述要求。一般来说，可以留下存储链的一部分不使用，这就是说，不能用任何区域确定这些部分，所以，K

和 M 的乘积不总是等于 N（但是仍然有 $KM \leqslant N$）的。换句话说，在存储链上的比特区域之间留下了“缝隙”。为了避免次级结构问题，建议使用只有嘧啶（或者嘌呤）形成的存储链和只有嘌呤（或者嘧啶）组成的粘贴链。

由于方法的多样性，可以利用其他可能的温度、盐浓度和化学溶解度来实现上述操作中每个步骤要求的亲和力。在这里，推荐使用自然产生的序列作为存储序列，因为这类存储序列容易批量生产，但前提是，这些序列满足以上的限制条件。

需要强调的是，以上略述的标准仅仅是为了作为例证，而一个更复杂的方法需要考虑到寡核苷酸配对次序所依赖的热力学参数。在系统设计中，为了防止错误粘贴链配对点可能的起泡失配，系统需要有一定的容错能力，而且必须能够阻止由于三螺旋结构造成的次级结构。而如果使用清除操作，将会引入额外的约束。

7.3 DNA 加密技术

7.3.1 使用随机一次性密码本的 DNA 密码系统

在 DNA 密码系统中，需要对输入的明文进行分割，使其成为具有固定长度的短的明文段。一次一密的加密方法是使用随机密码本将明文消息转换成加密的文本。关于一次性密码本的安全性有两点很关键：一是密码本必须确实是随机的；二是密码本只使用了一次。这种使用随机一次性密码本的密码系统是惟一一种被认为是绝对不可破译的密码系统。同样地，重复使用密码本就会增加被窃听器解密消息的风险。这两点原则规定了 DNA 序列所应具有的性质，下面会就这些性质进行进一步的讨论。首先以 DNA 链的形式秘密地建立随机的一次性密码本，并进行隔离和复制操作。进一步假设发送端和接收端预先共享特定的一次性密码本。这一假设需要初始时刻在发送端和接收端进行一次性密码本的传递，而溶液中 DNA 非常紧密的特性可以使得这一传递过程比较容易。下面将讨论如何创建这样的一种 DNA 一次性密码本。这里给出两种对大量的短消息序列进行加密的方法。这两种方法都是十分有效的，攻击者无法由生成的 DNA 中得到原始的明文消息。在这些加密方法中，一次性密码本是一种有效的随机密码本，它将短消息序列映射成加密序列（与固定密码本相反，在无约束的加密系统中，固定密码本不是一种机密的方法）。使用映射方式的加密方法有：①使用替换，即用一次性 DNA 密码本相应部分的相关匹配对每个消息序列加密；②通过生物分子计算技术进行异或计算。解密是用类似的方法。

7.3.2 使用替换的 DNA 加密系统

替换的 DNA 加密系统也就是一次性密码本的替代加密系统。其输入是长为 n 的被分割成固定长度的二进制明文消息。替代的一次性密码本由一张表组成，该表将所有可能的明文字符串映射为固定长度的密文字符，因此存在惟一对应的逆映射。在加密过程中，明文中的每个块被替代表中的密文字符取代，而解密则是反替代过程。在使用替代加密的情况下，希望以随机而可逆的方式，将试管中的 DNA 链（明文消息）转换成另一组完全不同的链（密文消息）。将明文加密为 DNA 密文链，同时将明文链移出。替代算法需要一次性密码本 DNA 序列来完成这种转换。该方法需要有许多小段构成的长的 DNA 密码本，每个小段又包

括密文字符及附加在后面的与其相应的明文。在明文转换成密文的第一步中，可以将字符对DNA链看作是一张查询表。理想的一次性密码本库是由大量的密码本组成的，每个密码本能提供完全惟一的随机的从明文到密文字符对的映射。

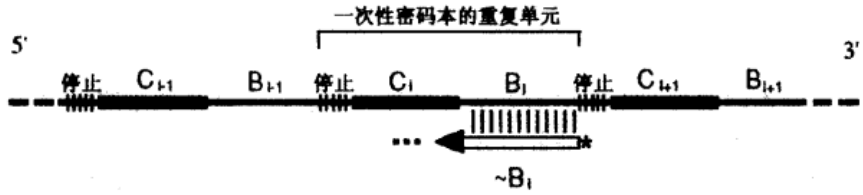


图 7-3　一次一密密码本 DNA 序列结构

图 7-3 给出了一个密码本结构的例子。重复的单元包括：密文或密码本匹配序列字符集的序列字符 B_i、明文集的序列字符 C_i，以及聚合酶“停止”序列。每个序列对 i 惟一地对应着一个明文字符及相应的密文字符。序列为 $\sim B_i$ 的 Oligo（具有 Watson-Crick 互补配对特性）可用作聚合酶引物，并可通过添加特定的明文字符 C_i 进行扩展。停止序列的作用是禁止增加的 DNA 链扩展超过成对明文字符的边界。每个来自该密码本库的链都指定了一个特殊的并且惟一的字符对的集合。一次性密码本由长度为 n 的 DNA 链组成，其包含 $d = n/(L_1 + L_2 + L_3)$ 个重复形式的副本，其中，密文字符长为 L_2，明文字符长度为 L_1，停止序列长度为 L_3（令 $L_1 = c_1 \log_2 n$，$L_2 = c_2 \log_2 n$，$L_3 = c_3$，其中整数常量 c_1，c_2，$c_3 > 1$）。每一个重复的单元都确定了一个单独的映射对，并且所有的密码本字符或明文字符都只能使用一次。因此，有密码本字符 B_i 与明文字符 C_i 的一一对应关系。停止序列作为“标点”出现在重复的单元之间，所以 DNA 聚合酶将不能持续地复制密码本链。停止序列由一系列相同的核苷酸组成，这些核苷酸通过减少 DNA 聚合酶的互补三磷酸酐来阻止链的复制。比如，如果聚合物缺少其基对的互补物，那么序列 TTTT 就会被视为停止点。这种序列的原型是由 Hagiya 提出的。基于这种结构，给引物退火，并以聚合酶扩充，产生一系列与明文或密文字符对链相关的寡核苷酸。字符对链的集合是随机密码本很重要的查询表。其可靠性依赖于词典的尺寸、可获得的密码本的数目、尺寸、复杂度和消息传送频率。表 7-1 可以用来评估查询表的风险。

表 7-1　简略计算表

参　数	范　围
词典尺寸	10000 ~ 250000 个字符
字符尺寸	8 ~ 24 个碱基对
密码本分集	106 ~ 108
消息尺寸	词典尺寸的（5 ~ 30）%

词典尺寸 d 指定了在单一的密码本里可以使用的字符的总数（如果使用 2D 芯片的输入输出，则 d 是指从 2D 芯片格中获得的像素数目）。字符尺寸 L_1、L_2 确定了一个词典可能得到的 DNA 序列的总数，但是随着互补序列的消除以及序列可能会分为两个词典，词典的尺寸就会相应地缩小。密码本分集表示了单一密码本建立中产生的随机密码本的总数。多次重复构造能产生大量惟一的密码本。假设随机密码本用于 DNA 芯片的 I/O 应用，则可以估计与词典尺寸相关的消息尺寸的范围。

关于密码本库的建立，可以借鉴用于基因构造过程中的分割组合或构造以及 DNA 计算中的 DNA 字符加密做法。明文和密文词典的序列字符的设计是面临的第一个技术挑战。就数学意义上来说，人们希望词典是不相交的。在实际应用中，还需要词典对于每一个密码本能够完整地覆盖，并且字符映射必须是惟一的（尽管完全不相关的词典是理想情况，但如果有少许重叠的词典，系统还是可以正常运行的）。

建立用于密码本的明文和密文对的方法有多种。方法之一就是利用溶液中随机组合的一次性密码本。因为很难达到完全覆盖并且还要避免明文或密文字符重复所可能产生的冲突，所以这种方法有一定缺陷（c_1 和 c_2 设置得足够大，就可以使在长度为 n 的密码本中出现重复字符的概率变得非常小，但这样覆盖率就降低了）。这些方法需要：设计两个独特的序列字符词典：其中一个词典用于密文字符，另一个词典用于明文字符。可以用正常的化学合成 DNA 的方法来产生这些词典，这一合成方法利用了在序列字符特定位置的序列随机性。在耦合反应中加入核苷酸磷就可以达到随机性的目的。因此，能确定所有有着某种复杂结构的可能序列库的合成。例如，RXXYRXRRYX（其中 R = A + G，X = A + C + G + Y，Y = C + T）会产生 16384 个可能的序列。这是一个以有偏随机合成形式生成的字符词典，此词典可用于序列空间受限的区域。为了建立两个词典，可以用两个有偏合成形式使词典重叠的程度最小化。为此，必须注意到这么一个事实：一个核苷酸基将会从明文字符编码里消失，尽管聚合会继续进行但会在遇到下面的停止序列时停止。如果增加密文字典中相同碱基的比例，那么自然就会减少公用的字符序列。虽然完全不相交的词典是理想的，但系统在有少许可能的词典重叠时仍可运行。合成寡核苷酸包括三个部分：第一部分是按照密文字符的有偏形式，第二部分是决定明文字符的形式，第三部分是恒定的停止序列。合成之后，寡核苷酸将会连接在一起，并克隆成适当的矢量。在克隆和传输细胞后，每一个单独的克隆将产生一个独一无二的随机的替代密码本，所以需要的密码本的数目仅取决于细菌副本的个数。

建议采用的办法是使用 DNA 芯片。Affymatrix 使用了包含被 MeNPOC 修改后的磷化学性质，而不是用 DMTr。芯片阵列中，每个像素占 $10\mu m^2$，因此，完整的 12 个碱基序列的阵列约占 $4cm^2$。当使用芯片来显示 CGNNNNCG 序列可能的全部词库时，首先通过屏蔽闪光取消对所选区域的保护，然后连接承载 MeNPOC 保护组的单体基。循环即可在新的位置重复进行。目前这样的 DNA 芯片是可以买到的，并且构造常规变体的化学方法也已经比较成熟。DNA 芯片阵列是固定的 DNA 链，所以单一序列的多个副本可以组合在一个微小的像素内。DNA 芯片的这一微探针阵列在光学上是可寻址的，并且已知的技术可以在阵列的每个点产生独特的 DNA 序列。光引导聚合可以使成千上万的位置并行进行 DNA 合成的化学反应。因此，准备的序列的总数远远超过所需要的化学反应的总数。为了准备长度为 L 的寡核苷酸，需要在 $4n$ 个化学反应中合成 $4L$ 个序列。例如，要构造明文/密文对，65000 个长度为 8 的序列需要 32 次合成循环，1.67×10^7 个长度为 10 的序列仅需要 48 次循环。由此生成的每个密码本的词典接近完全覆盖，而且明文和密文对之间也会有接近惟一的字符映射。借助很多已知的技术，这些密文字符和明文字符能随机地组合在一起，通过克隆或 PCR 技术可扩大合成的一次性密码本。

7.3.3 DNA 异或一次性密码本密码系统

在传统的密码学中，Vernam 密码是这样实现的：首先有一个独立随机分步的比特序列

R，然后由此生成一个序列 S 作为一次性密码本。复制一次性密码本，并给发送方和接收方分别发送一份。令 L 等于 S 中还未使用的比特数，即初始时 $L=R$。XOR 是一种运算操作，对于给定的两个逻辑输入，如果输入是相同的，则输出 0；如果输入是不同的，则输出 1。当需要发送的二进制明文消息的长度为 $n<L$ 时，每一比特 M_i 与比特 $K_i=SR-L+i$ 进行异或，可产生加密的比特 $C_i=M_i\text{XOR}K_i$，这里 i=1，…，n。S 中已使用的 n 个比特在发送端和接收端就被丢弃，而加密后的序列 $C=(C_1, C_2, \cdots, C_n)$ 被发送到接收端。在接收端，重复同样的过程，即序列 C 与 S 进行逐位比特异或的运算，用完之后丢弃 S 的比特。因为 $C_i\text{XOR}K_i=M_i$，$C_i=M_i\text{XOR}K_i$，并且 $M_i\text{XOR}K_i\text{XOR}K_i=M_i$，异或可交换的性质导致初始消息的复制（对于有效的 DNA 编码，是以四进制的方式进行运算的。因为 DNA 有 4 个核苷酸碱基，所以 DNA 编码与二进制不同，它使用的是模 4 运算。在这种情况下，假设输入明文和一次性密码本是四进制的，则加密就需要使用模 4 运算而不是异或运算，对于解密，需要用密文减去一次性密码本的元素模 4 后的值。为简便起见，仍以二进制的形式讨论）。我们期望以随机而可逆的方式，将试管中的短 DNA 链（明文消息）转换为完全不同的链（密文消息）。假设每个明文消息附加有固定长度为 L_0 的前缀索引标志，每个一次一密的 DNA 序列也有附加的惟一的前缀索引标志，其长度同样为 L_0，这是对明文消息标志的补充。凭借已知的 DNA 重组技术（如退火和粘接），明文消息中每个相关的对和一次一密序列有着相同的标志，因此它们可以粘接成一个单独的 DNA 链。DNA 编码消息可以用逐位比特的异或运算来修改，所以使用一次一密 DNA 序列，明文消息的分段可以转换为密文链。解密的过程是类似的，利用的是逐位比特异或运算的可交换性质。显然，需要寻找一种使向量也可以进行异或的运算方法。一些已知的二进制操作算法和逐位比特异或运算是相似的，但却在需要处理进位比特时有额外的限制。如果有一种方法可以进行加法运算，那么就可以在忽略进位比特的情况下，用这种方法来稍微改变映射关系以反映逐位比特异或的规则，而不使用那些考虑进位比特的比特加法。要获得所需的逐位比特异或的计算方法，可对之前两种已知的生物分子计算技术中的整数加法修改后得到。

Guarnieri，Fliss 和 Bancroft 在重组 DNA 中首次建立了 BMC 加法操作的原型。虽然这一试验工作具有非常重大的意义，但是它仍然有某些局限性：①只能对两个数字进行加法运算。所以它并没有用到 BMC 巨大的并行处理的能力。②输出很显然是从输入编码而来，因此它并不允许有重复的操作。他们随后提出了一些基本的运算方法，如加法和减法等。这些算法允许对粘接操作的输出和输入进行更大规模的并行操作。Rubin 等人对 BMC 方法用于粘接整数算法进行了试验性的论证。这也是首次通过有限异或运算对 BMC 逻辑可逆计算的论证。这一方法可以直接用于向量异或运算。

另一个已知的可有效地进行一般的二进制加法，尤其是可有效地进行异或运算的方法是利用 DNA 分子瓦的结构。他们的思路是基于 Winfree 等人关于在 DNA 自组方面的研究结果以及 Reif 压缩组合方法。为了有效地自我组合，需要首先建立有着特殊的无补体的密码本的分子瓦。分子瓦的原型是由 LaBean 在实验室中建立的。设计这样的分子瓦的目的是基于从代表输入的分子瓦组合创建的初始态，计算的输出可以实现自我重组。更为特别的是，考虑到二进制输入串，一个单独的分子瓦就代表了一个比特。这样设计的分子瓦是为了能够以线性组合代表二进制串。特殊的角分子瓦的使用可以让两个代表二进制输入串的线性分子瓦组合联合到一起，并且可以创造一个紧密的框架，在这一框架内，输出分子瓦能够很好地组

合在一起。LaBean 在这一过程中考虑了无中介的二进制加法或异或。在流程的最后，生成了一个贯穿整个组合的包括两个输入和输出的单独的链。利用这个性质，就可以实现 DNA 的 Vernam 密码。

下面对如何进行逐位比特异或运算作简要介绍，如图 7-4 所示。对于消息的每个比特 M_i，构造了一个序列 a_i，它代表了一个长的输入序列的比特。通过利用适当的粘接序列，将消息 M 的 n 比特组合为一个序列 $a_1a_2\cdots a_n$ 作为每个二进制输入的染色体支架链。另有一部分染色体支架链 $a_1'a_2'\cdots a_n'$ 建立在随机输入的基础上，并作为一次性密码本。设想许多最初由 PCR 或适当的技术来创建和克隆的形式如 $a_1'a_2'\cdots a_n'$ 的染色体支架，在发送端和接收端进行分离和存储。当发送端需要加密时，通过引入消息染色体支架、一次性密码本染色体支架、角分子瓦和各种不同的用来完备分子瓦的序列，以及收发端都知道的前缀索引标志等来传递染色体支架。创建输入染色体支架和在支架上组合分子瓦的过程，已由 LaBean 在实验室中成功地实现。最终，加法粘接酶产生了连续的报告链 $R = a_1a_2\cdots a_n \parallel a_1'a_2'\cdots a_n'. \parallel b_1b_2\cdots b_n$（这里 $b_i = a_i\mathrm{XOR}a_i'$，$i=1, 2, \cdots, n$ 符号“$\parallel$”表示两个链串联），它贯穿在整个组装过程中。可以通过溶解和净化分子瓦的小序列来提取这一包括了输入消息、密钥和明文的报告链。其步骤是：首先，利用标志序列，根据密文的长度将密文分离；然后存储到压缩表再传送到目的地。XOR 使得与密钥相关联的 Vernam 密码可逆。特别地，当 $b_1b_2\cdots b_n$ 用作输入染色体支架时，根据索引指示，另一个染色体支架来自于存储的 $a_1'a_2'\cdots a_n'$。除此之外，常用的分子瓦重建序列还需加入角分子瓦和粘接酶。在自我组合之后，报告链就被溶解、净化，在标志处断开，这样明文就提取出来了。可以注意到，分子瓦的建立中引入了容错手段。

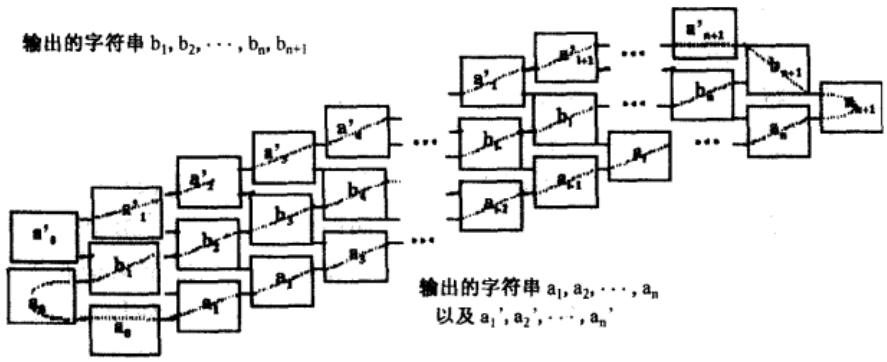


图 7-4　通过 *DNA* 瓦进行的 *XOR* 运算

7.4　DNA 加密技术的应用

7.4.1　二维图像 DNA 密码系统

本节将通过一个加密/解密系统的例子对 DNA 密码系统进行概要说明。该系统的输入和输出的数据是记录在 DNA 芯片微探针阵列上的二维图像。这一系统包括：需要加密的数据集合、承载着固定 DNA 链的芯片和用于加密 DNA 长链的一次性密码本。在该系统中，需要

加密的数据集是二维图像，但通过对这一方法的修改，它同样可以应用于加密和解密其他形式的数据或信息。由于 DNA 芯片包含可寻址的核苷酸序列阵列，所以一个单独的序列的多个复制可以组合在一起，形成一个微小像素。这样的 DNA 芯片目前已经可以买到了，而且可以根据不同的用户要求采用不同的化学工艺。下面与芯片相关的一些信息。

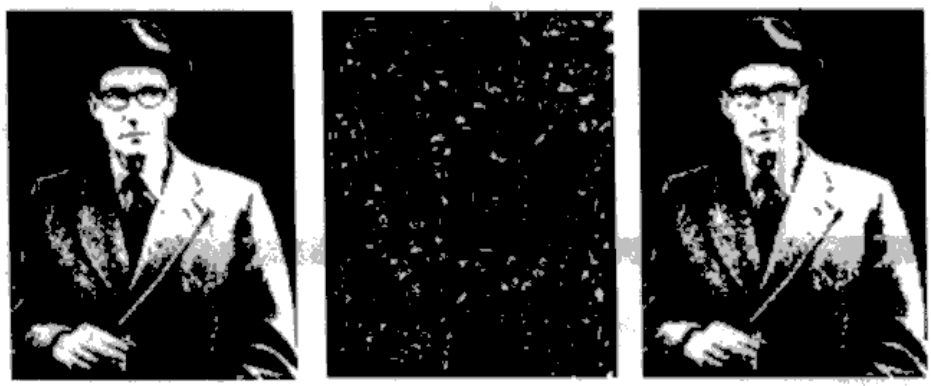


图 7-5　从 DNA 芯片的 I/O 口读取的模拟

图 7-5 给出了大致的 I/O 方法的显示。荧光标记的字符对 DNA 链来自替代的一次性密码本。在 DNA 芯片特殊的位置（像素）进行退火操作，得到它们的补体序列。消息传送到一个有着透明的和不透明区域的挡光板下。随着挡光板保护芯片的闪光，在透明挡光板像素下退火的寡核苷酸在光不稳定的位置分解，其 5’ 部分从退火的 3 部分分离（见图 7-3），并进入溶液中。此试管中的 DNA 链便是加密后的消息。在不透明挡光板下退火的低聚物不会受到闪光的影响，因此可从芯片上剥离出来并丢弃。如果已加密的消息低聚物重新退火到 DNA 芯片上，它们将退火到一个新的特殊的位置，消息将无法阅读。将荧光标记的低聚物作为引物可以用来解密。当字符对 PCR 产物都限定在相同的 DNA 芯片中时，解密的消息就可以显示了。

图 7-6 中，退火 DNA 与来自随机替代密码本的字符对链相一致。稳定的 DNA 链定位于芯片的玻璃支持物上。退火链包括：在 5’ 末端的荧光标记，匹配密码本序列字符，可分离 DNA 主干的光不稳定基和芯片匹配字符。5’ 末端携带着密文字符，而 3’ 末端携带明文字符。在两个序列字符之间包含了一个可分离感光基的类似物。

加密的步骤如下：

1）荧光标记的字符对链退火到芯片上承载明文补体的像素 3’ 末端。

2）挡光板阻止闪光照到某些像素。在非保护的区域，DNA 主干在明文和密文字符之间的地方分离。

3）密文字符链在它们的 5’ 末端也由荧光标记，然后作为加密消息的收集和传送工具。

在接收端，只有使用相同的一次性密码本和与加密过程中相同的 DNA 芯片，消息才可被解密。首先，字符对链必须用合适的明文字符附加到密文字符上才可重建。聚合酶扩展或不均匀的 PCR 把密文字符用作引物和一次性密码本可达到重建的目的。在解密过程中，仍然需要荧光标记，但此时不一定需要光不稳定基。解密的最后一步是将修改后的字符对链绑定到 DNA 芯片上，并由荧光显微技术读出消息。图 7-7 是加/解密中各个步骤的图示。

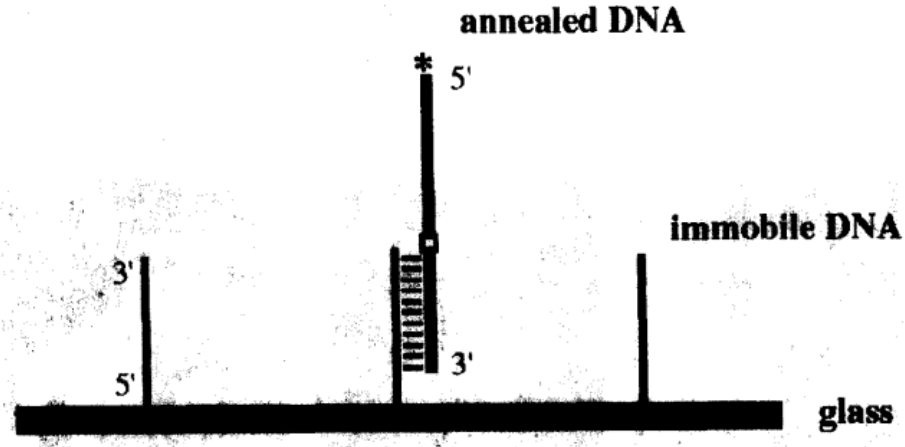


图 7-6　DNA 芯片的成份及组成

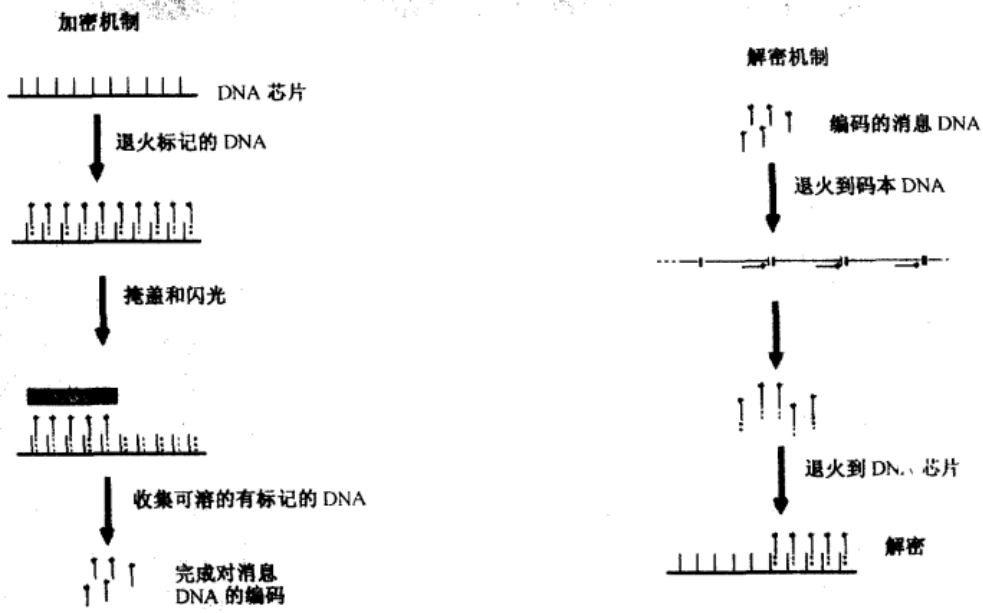


图 7-7　加密和解密的步骤

7.4.2　DNA 隐写术

隐写术是一种将秘密信息隐藏到其他信息中的技术。在隐写术系统中，初始明文并不用加密，而是伪装或隐藏在其他的数据中。已有的一些隐写术系统使用网格来显示图像中除秘密消息以外的所有信息，有的系统在大型图像中隐入微型图片，有的系统使用不可见的水印等等。密码学文献一般认为传统的隐写技术的安全性比较低，并且还有许多实际中隐写技术被破译的例子。但是，由于隐写术的简易性，使得人们还是非常关注隐写术。很多技术把隐写术应用到生物分子计算的内容中。其方法之一就是输入一个或多个 DNA 链（即看作是明文消息），然后为其附加一个或多个随机构造的“密钥”链。将产生的“标记明文”DNA 链与其他很多“混淆”DNA 链相混合，以达到隐藏明文的目的。同时这些用于“混淆”的

DNA 链也可能是随机组合而成的。根据“密钥”链的知识，凭借很多已知的重建 DNA 的分离技术，可以解密产生的 DNA 链的溶液。例如，明文消息链可以通过杂交而分离出来。

1967 年，Kahn 对隐写的概念进行了定义：隐写可以看作是将秘密值隐藏在别的信息中间以掩盖其存在性的方法。在密码学上，隐写被认为是一种简单的密码方法。DNA 计算最近被用作隐写方法是由 Clelland 等人在 1999 年提出来的。Clelland 等人将秘密信息进行编码作为 DNA 的链并隐藏于众多的随机 DNA 中。

就像数字消息经常用 1 和 0 的序列表示一样，二进制 DNA 表示也得到了应用。二进制编码非常适用于数据结构以及简单和快速的解密系统。解密可以由适用于数字 DNA 的方法来完成。通过这种方法，信息内容可以被 PCR 和电泳直接解密和读取。

DNA 二进制链通过短的双链 DNA 分子串联而成。因为 DNA 分子含有重叠的序列，因此，需要通过退火和粘贴作用聚合为 DNA 二进制链，如图 7-8 所示。为了将单一的分子从所有的 DNA 链里分离出来，分子被结扎成为粒体并以细菌的方式复制，如图 7-9b 所示。那么通过 PCR 和电泳就可以读出每一个被复制的链的信息内容。

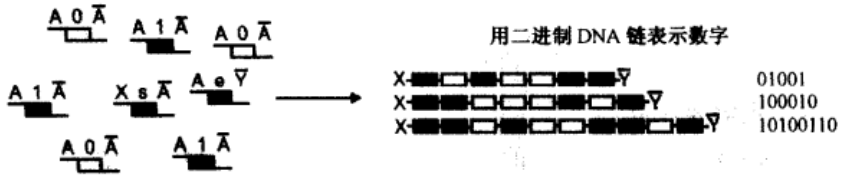


图 7-8　DNA 二进制链的组合

这里，所有的二进制链具有 s|o|i|e 的形式，即二进制链由两个结束符 s(Start) 和 e(End) 和在两个结束符之间的 DNA 的一个任意数字构成。串联操作是由退火和结扎过程完成的。结束符和 DNA 比特由退火的寡核苷酸构成。在寡核苷酸的两边是有粘性的用于串联的末端（A，$\overline{A}$，X，$\overline{Y}$）。粘性端 A（$\overline{A}$）起到正确链接比特和结束符的作用。这里 X 和 Y 是进行复制所需要的粘性端。

1. DNA 隐写

作为读出过程，如果没有别的方式来读取二进制链，那么引物序列就是读取二进制链所必须的了。通过将某个二进制链和其他的 DNA 混合在一起，就可以作为一种隐写的方法。

需要说明的是，这种混合的数量是非常大的，其数量等于每克物质所含的分子数。为了达到更好的安全性，“混淆”链需要有与信息链相同的二进制形式。对于解密，需要用到附着在信息链上的惟一的鉴定序列（密钥序列）。这个序列可以是任何结束符序列，但是更一般的是开始序列（参看图 7-9）。这样，通过将适当的密钥序列作为 PCR 反应的引物，就可以解密读出消息链。

2. 安全性

在对加密的安全性分析中，通常假设明文（以消息链给出）具有所描述的二进制结构，并且按照上面所提到的过程进行加密和解密。下面是对通信协议的描述，假设有两个参与者 A（Alice）和 B（Bob）。

1）A 和 B 通过安全信道交换生成密钥。

2）A 根据二进制配对生成消息并且在一个结扎过程中加入一个密钥。

3）A 生成一定数量的“混淆”DNA，并且将消息链放于这些“混淆”DNA 中。

4）A 经由一个开放的信道将结果发送给 B。

5）B 解密消息。

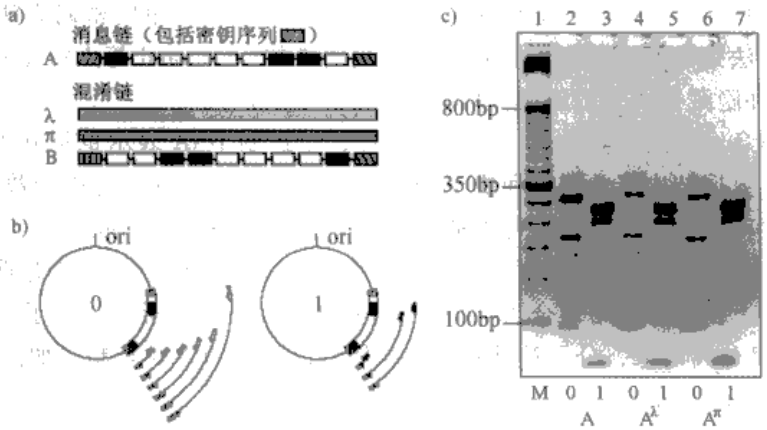


图 7-9　用 DNA 二进制链做的隐写

在这里，有 3 点需要注意：

1）含有惟一的密钥序列（Start）的消息链 A 可以隐藏在各种“混淆”DNA 链中。这些“混淆”链可以是随机的 DNA，比如抗菌素 λ（λ）或者鲭鱼精子 DNA（π），或者是不同密钥序列的 DNA 二进制链（B）。注意，λ、π 和 B 代表一个由大量的互不相同的“混淆”链构成的集合。

2）A 的解密框架。对于接收方，只有在密钥序列已知的情况下才可以解密，这是因为 PCR 所读出的是基于两个引物的信息。在这里，默认的 DNA 比特序列是公共序列，而结束符序列被认为是秘密的。

3）消息链 A 的解密。在图 7-9 中，第 2、3 列给出的是未加密的 A，第 4、5 列给出的是由 λDNA 加密的 A 的解密，第 6、7 列给出的是由鲭鱼精子 DNA（π）加密的 A 的解密。用于加密的 DNA 具有两个条件：具有克分子数相等的 A 类的消息链；加密的 DNA 在读出过程中不干扰引物。

这项技术的安全性基于这样一个概念：截获者在没有特征或者标记的情况下，无法区分“混淆”链和消息链，因此，为了得到消息链，截获者需要寻遍整个空间。换句话说，只有在已知密钥相关信息的情况下，截获者才能够成功地解密。任何单一的攻击一定不能产生比随机抽取一个链获得的信息更多。为了达到这一安全性要求，必须要考虑两个方面：第一，每一个“混淆”链都需要有和消息链同样的结构，也就是不可分；第二，对消息的 DNA 编码可以抗语言统计攻击。这里用于“混淆”的链是随机的基对序列，并且明文来源于自然语言比如英语，消息链原始的明文部分是可以从“混淆”链中区分的。截获者可以利用这个优势，通过用区分标志的方法来进行分离以重复减少消息链的数量与“混淆”链的数量的比率。这里所描述的密码系统所用的“混淆”链都是由 DNA 二进制链构成。此类二进制链由长为 L 的序列串联而成，并且具有和消息链的密钥序列相同的长度。DNA 二进制链的长度也与消息链的长度相对应。而且，设 D 是不同的“混淆”链的集合，$d = |D|$ 是其元

素的数目。为了抗语言统计分析，系统需要保证消息的 DNA 编码中 0 和 1 具有相同的发生概率。假设消息可以以一种抗语言统计分析的形式来编码。对于密码分析，有如下假设：

1）截获者具有和发送者及接收者相同的技术能力和资源。

2）截获者知道消息的发送并且可以接入开放的信道。

3）截获者知道密码系统（特别是加密方法）。

4）分子的加密方案不能复制。这样如果截获者希望隐藏攻击，那么他/她就需要有完备的复制溶液的能力。

5）截获者不知道消息链的密钥。

6）每一个密钥发生的概率相同，但这并不意味着每一个序列一定会发生。

7）所有的 DNA 链在溶液中随机、均匀地分布。

8）每一个链单个出现的频率不能反应出它是否是消息链。同时，所有的链具有相同的出现频率或者完全随机。

9）截获者如果知道了密钥序列，他就可以过滤所有的“混淆”的链。因此，对于密码分析，不再需要考虑单个链的出现频率问题。

对截获者来说，他区分“混淆”链和消息链的概率是很小的。惟一的方法是随机地抽取或者猜测密钥序列。如果随机选择消息链的概率是 $1-\sigma$，那么这个方法的安全性就是 σ $(0\leqslant\sigma\leqslant1)$。基于以上阐述，截获者获得消息链的概率越小，他攻破这个系统所付出的努力就越大，那么这个系统就越安全。

（1）系统安全性（情况 A）

设 S_E 是指定的密钥序列，S_T 是任意的序列，$P(S_T \| S_E)$ 是 S_T 和 S_E 绑定在一起的可能性。另设序列的长度为 L，碱基种类为 $B=\{A, T, G, C\}$ $(|B|=4)$，则 $P(S_T \| S_E)$ 的概率为：

$$P(S_T \| S_E) = \frac{1}{|B|^L}$$

只有 S_T 在每一个单一的基上是 S_E 的补的情况下，这个结论才正确，即它们的汉明距等于 0（$h(S_T, S_E)=0$）的情况下才成立。但是，还需要考虑在非完全补序列之间发生的退火的情况。在此情况下，汉明距为 i 的序列 S_T 的数量是：

$$\binom{L}{i}(|B|-1)^i$$

取尺寸为 d 的“混淆”链的集合，设 $\lceil H(d, L)\rceil$ 是错误引物数目的最大值，在此情况下，仍然可能存在明确的绑定。因此有概率：

$$P(S_T \| S_E) = \frac{1}{|B|^L}\sum_{i=0}^{\lceil H(d,L)\rceil}\binom{L}{i}(|B|-1)^i$$

这里相加得到的和就是生成序列 S_T 的数目，S_T 满足 $h(S_T, S_E)\leqslant\lceil H(d, L)\rceil$。通常 $H(d, L)$ 不是一个整数，因此这里 $\lceil H(d, L)\rceil$ 代表最坏的情况。S_E 是否未知并不重要，因为这个值对于所有的 S_E 序列来说都是一样的。总而言之，安全性 σ_1 就变为：

$$\sigma_1(d,L) = 1-\frac{1}{|B|^L}\sum_{i=0}^{\lceil H(d,L)\rceil}\binom{L}{i}(|B|-1)^i$$

为了在这一点上得到一个特殊的值，就需要确定 L 和 $H(d, L)$。

$H(d, L)$ 是指定的允许任意的 S_T 经过明确的退火过程得到密钥序列 S_E 的错误引物的最大数目。这也就是说，S_T 到 S_E 的汉明距离也许并不比 S_T 到 d 个“混淆”序列集合的最大的汉明距离小。更直观地说，只要 S_T 和 S_E 之间的汉明距小于 S_T 和所有伪序列 S_i（$i=1, 2, \cdots, d$）的汉明距离，那么 S_T 就与 S_E 存在绑定关系。可以将此概念扩展到“混淆”序列中，其与 S_T 的汉明距离等于 S_T 到 S_E 的汉明距离的情况。这种情况使得系统的安全性处在最差的条件下。因为在这种情况下，如果由于“混淆”序列与 S_E 的汉明距离的原因使得“混淆”序列不能被正确识别的话，S_T 也会退火为 S_E。实际上，明确的汉明距离是不可计算的，因为“混淆”序列集中的序列和密钥序列是随机序列。因此，需要考虑 $H(d, L)$ 是所有可能的链集合（尺寸为 d 的所有 $|B|^L$ 可能序列的子集合）中的 S_i 和 S_E 之间的平均最小汉明距离。平均最小汉明距离 $\min_{S\in D}\{h(S_E, S)\}$ 与参考序列 S_E 是独立的，因此有 $h(S) := h(S_E, S)$。尺寸为 $|B|^L-1$ 的集合中尺寸为 d 的子集合的数目为 $\binom{|B|^{L-1}}{d}$ 的参考序列一定不在任何集合 D 中。D 满足下列条件：

$$H(d,L)=\frac{1}{\binom{|B|^{L-1}}{d}}\sum_{D,|D|=d}\min_{S\in D}\{h(s)\}$$

需要注意的是，所有子集合的最小的测定都可以明确给出，因此，有下面的表述：

尺寸为 $|B|^L$ 的集合的 d 尺寸的子集数目等于：

$$\binom{|B|^L}{d}$$

含有特定序列的尺寸为 d 的子集数目为：

$$\binom{|B|^L-1}{d-1}$$

在尺寸为 n 的集合中，含有至少一个特定序列的尺寸为 d 的子集数目是：

$$\sum_{i=1}^{n}\binom{|B|^L-i}{d-1}$$

至少含有 n 个给定序列中的一个而不含有 m 个给定序列中的任何一个序列的尺寸为 d 的子集数目等于：

$$\sum_{i=1}^{n}\binom{|B|^L-i-m}{d-1}$$

综上所述，有：

$$H(d,L)=\frac{1}{\binom{|B|^{L-1}}{d}}\sum_{i=1}^{L} i\sum_{j=1}^{\binom{L}{i}(|B|-1)^i}\binom{|B|^L-a(i,L)-j}{d-1}$$

这里 $a(i,L)=\sum_{j=0}^{i=1}\binom{L}{j}(|B|-1)^j$，即 $a(i, L)$ 是汉明距离小于 i 的序列的数量。第一个求和考虑到所有最小汉明距离为 i 的子集，即子集含有至少一个汉明距离为 i 的序列。每一个

加数至少含有$\binom{L}{i}$（$|B|-1)^i$个给定序列中的一个而不含有a（i，L）个给定序列中的任何一个序列的尺寸为d的子集的数目。

为了简化，上式可以写为：

$$H(d,L)=\frac{1}{\binom{|B|^L-1}{d}}\sum_{i=1}^{L}\binom{|B|^L-a(i,L)}{d}=\sum_{i=1}^{L}\prod_{j=0}^{a(i,L)-2}\frac{|B|^L-d-j-1}{|B|^L-j-1}$$

对于不同的密钥长度L和集合D的尺寸d，表7-2给出了与「H（d，L）Ł相对应的一些σ_1（d，L）的值。

表7-2　安全值σ_1

L/d	1	100	$2^{10}-1$	$|B|^L/2$	$|B|^L-1$
5	0.23730 (4)	0.89648 (2)	0.98438 (2)	0.89648 (2)	0.98438 (1)
10	0.24403 (8)	0.98027 (4)	0.99649 (3)	0.99958 (2)	0.99997 (1)
20	0.22515 (16)	0.98614 (10)	0.99606 (9)	0.99999 (2)	0.99999 (1)

（2）系统安全性（情况B）

如果截获者可以将特定的链从溶液中分离出来，那么他/她就不必合成一个任意的DNA序列。而且，截获者可以将独立出来的链进行排序，并且直接读出链。这样，隐写系统安全性将直接依赖于结果中不同的链的数目。在结果中，共有$d+1$个不同的链（“混淆”链+消息链）。碰巧分离出消息链的概率如下：

$$P(\text{碰巧分离出消息链的概率})=\frac{1}{d+1}$$

因此，安全系数σ_2就变为：

$$\sigma_2(d)=1-\frac{1}{d+1}$$

对于这两种攻击方法，分别对应有安全值σ_1和σ_2。考虑到特殊情形，即“混淆”链集合D是最大尺寸$|B|^L-1$的情形，情况A和B是相同的，那么两个安全值就相等，即对于任意L和$d=|B|^L-1$：

$$\sigma=\sigma_1(d,L)=\sigma_2(d)=1-\frac{1}{d+1}$$

7.5　展望

DNA密码的研究还处于探索阶段，准确预测其未来的发展还不太现实。考虑到生物学技术的发展以及密码学的需求，对今后DNA密码的发展提出如下建议：

第一，实现DNA密码要以现代生物学为工具，以生物学难题作为主要安全依据，充分

发掘 DNA 密码独特的优势。加密和解密的过程就是对数据进行变换的过程。在电子计算机和互联网高速发展的今天，这些变换过程如果能够用数学工具来描述，常常比物理或者化学变换要容易实现。其他类型的密码系统能够存在，就应该具备比现有的密码系统更好的安全性或者更高的数据密度等特性，不能或者不易用数学方法和电子计算机来实现。DNA 密码要能够存在和发展，就必须充分挖掘 DNA 密码的优势，利用 DNA 所具有的体积小的特性，实现纳米级的存储；利用 DNA 的超大规模并行特性，实现加密和解密的快速化；利用人类还不能破译但是能够利用的生物学难题，作为 DNA 密码的安全依据，以实现能够抵抗量子攻击的新型密码系统。由于目前还无法完全确定量子计算机对各种数学难题的威胁，DNA 密码的安全依据也不应完全排斥数学难题。考虑到 DNA 计算的巨大并行能力，用电子计算机难以实现的加密和解密也许可以用 DNA 计算机轻松实现。如果这种算法能够抵御量子计算机的攻击，这种计算的安全性就可以用于 DNA 密码中。所以，DNA 密码与传统的密码并不是完全互相排斥的，有可能把二者结合起来形成混合的密码系统。

第二，安全性要求。不管 DNA 密码和传统的密码有多么不同，它同样要遵守密码的共同特性。DNA 密码的通信模型仍然是由发送者和接收者组成的，双方通过安全的或认证的途径获得密钥，然后通过不安全或非认证的途径传递密文进行通信。对于 DNA 密码的安全性要求也仍然应该以 Kerckhoff 提出的假设为依据：一个密码系统的安全性必须仅依赖其解密密钥，即在一个密码系统中除解密密钥外，其余的加/解密算法等均应为假设破译者完全知道。只有在这个假设下，破译者仍然无法破解密码系统，此系统方有可能称为安全。具体地说，就是假定破译者知道密码的设计者所用的基本生物学方法，并且有足够的知识和精良的实验室设备能够重复设计者的操作。破译者所不知道的，只有密钥。在 DNA 密码中，密钥通常是某些生物学材料的实物、制备流程，或者实验条件等。

第三，DNA 密码现阶段的研究目标是以安全且容易实现为主，存储密度为次。一个好的密码系统，既要是安全的，还要是容易实现的。现代生物技术的发展，使得科学家可以用 DNA 来表达数据。但是，这方面的技术还是刚刚起步。目前，对只有纳米级的 DNA 直接操作很困难，只能通过 PCR 等生物学扩增技术，把 DNA 序列大量扩增后，在各种限制性酶的帮助下操控 DNA 序列。在当前的技术水平下，还远远不能用几克 DNA 存储世界上所有的数据。如果一味追求存储密度的提高，就难以在现有的技术水平下实现 DNA 密码。现阶段，把大量的 DNA 所表现出来的群体性质用于密码学更实际一些。比如，采用 DNA 芯片来储存数据并用杂交来读取数据，可以实现快速、方便的数据输入/输出。虽然 DNA 芯片的数据存储密度要比直接用核苷酸编码的方法低，但是实现起来要容易得多。

第四，DNA 密码现阶段的主要任务是建立起 DNA 密码的基本理论依据，积累研制 DNA 密码的实际经验。

可以肯定的是，DNA 具有用于高密度数据存储和超大规模并行计算的潜力，这是研究 DNA 计算和 DNA 密码的动力。现在的目标和难点是如何充分挖掘和利用这些潜力，这方面的研究在国际上也是刚刚起步。无论是 DNA 计算，还是 DNA 密码，都还没有建立起完善的理论。现代生物学仍然偏重于实验而不是偏重于理论。一个 DNA 密码系统的安全性所依据的生物学难题有多么难，相应的密码系统有多么安全，这些都还没有有效的方法来衡量。如何建立起类似于计算复杂度理论的方法来评估生物学难题的难度，是一个迫切需要解决的问题。所以，现阶段最重要的工作是发掘 DNA 可用于计算和加密方面的优良特性，建立起

DNA 密码的理论依据并积累 DNA 密码的研制经验，为研制安全且方便实用的 DNA 密码系统打下基础。

7.6 小结

本章首先描述了与 DNA 密码系统密切相关的粘贴模型，对基于此模型的信息表述、串集合的操作及物理实现进行了较为详细的阐述。DNA 密码是一个新兴的研究领域，目前成熟的密码系统还不多，所以，本章仅对 3 种 DNA 密码系统：使用随机一次性密码本的 DNA 密码系统，使用替换的 DNA 加密系统，DNA 异或一次性密码本密码系统进行了描述，并在接下来的部分给出了一个使用 DNA 芯片和随机组合的一次性密码本的二维图像 DNA 密码系统的例子和一个隐写的例子。

目前，DNA 密码处于研究的初期，还有很多问题有待解决。但是 DNA 分子所固有的超大规模并行性、超低的能量消耗和超高的存储密度，使得 DNA 密码能够具有传统的密码系统所不具有的独特优势。

7.7 习题

1. 说明基于 DNA 的一次一密的工作原理。
2. 阐述 DNA 密钥的提取方法。
3. 论述 DNA 密码技术的优点以及目前存在的主要问题。
4. 比较 DNA 隐写技术与数字水印技术的异同。
5. 综述影响 DNA 密码安全性的因素。

附录 缩略语

缩略语	英文	中文
AES	Advanced Encryption Standard	先进加密标准
BMC	Biomolecular Computing	生物分子计算
CBC	Cipher Block Chaining	密码分组链接模式
CFB	Cipher Feedback	密码反馈模式
CTAK	Cipher Text Auto Key	密文自动密钥
DES	Data Encryption Standard	数据加密标准
DLP	Discrete Logarithm Problem	离散对数问题
DNG	Deoxynucleic guanidine	脱氧核酸胍
ECB	Electronic Code Book	电子密码本模式
ECC	Elliptic Curve Cryptography	椭圆曲线密码
ECDH	Elliptic Curve Diffie-Hellman	基于椭圆曲线的 Diffie-Hellman 协商
ECDLP	Elliptic Curve Discrete Logarithm Problem	椭圆曲线离散对数问题
ECDSA	Elliptic Curve Digital Signature Algorithm	椭圆曲线数字签名算法
KAK	Key Auto Key	密钥自动密钥
LCG	Linear Congruential Generator	线性同余随机数产生器
LFSR	Linear Feedback Shift Register	线性反馈移位寄存器
NLFSR	Non-Linear Feedback Shift Register	非线性反馈移位寄存器
OFB	Output Feedback	输出反馈模式
PC	Personal Computer	个人计算机
PCR	Polymerase Chain Reaction	聚合酶链反应
PKC	Public-key Cryptography	公钥密码学
PNA	Peptide Nucleic Acid	肽核酸
PRBS	Pseudorandom Binary Sequence	伪随机二值序列

参考文献

[1] William Stallings. 密码编码学与网络安全:原理与实践[M]. 杨明,胥光辉,齐望东,等译. 2版. 北京:电子工业出版社,2001.

[2] 王育民,刘建伟. 通信网的安全——理论与技术[M]. 西安:西安电子科技大学出版社,2000.

[3] 卿斯汉. 密码学与计算机网络安全[M]. 北京:清华大学出版社,2001.

[4] Paul Garrett. 密码学导引[M]. 吴世忠,宋晓龙,郭涛,等译. 北京:机械工业出版社,2003.

[5] Michael Welschenbach. 密码编码学——加密方法的C与C++实现[M]. 赵振江,连国卿,等译. 北京:电子工业出版社,2003.

[6] Joan Daemen, VincentRijmen. 高级加密标准(AES)算法——Rijndael的设计[M]. 谷大武,徐胜波,译. 北京:清华大学出版社,2003.

[7] Carolynn Burwick, Don Coppersmith, Edward D'Avignon, et al. MARS - a candidate cipher for AES (Revised) [R]. IBM submission to AES, 1999. http://domino. research. ibm. com/comm/research_projects. nsf/pages/security. mars. html.

[8] Ronald L. Rivest, M. J. B. Robshaw, R. Sidney, Y. L. Yin. The RC6™ block cipher[R]. RSA submission to AES, 1998. http://theory. lcs. mit. edu/~rivest/rc6. pdf.

[9] Bruce Schneier, John Kelsey, Doug Whiting, etal. Two fish: A 128 bit block bipher[R]. Counterpane Labs submission to AES, 1998. http://www. schneier. com/paper-twofish-paper. html.

[10] Ross Anderson, Eli Biham, Lars Knudsen. Serpent: A Proposal for the Advanced Encryption Standard[R]. AES proposal, 1998. http://citeseer, ist. psu. edu/253207. html.

[11] Joan Daemen, Vincent Rijmen. AES Proposal: Rijndael[R]. AES proposal, 1998. http://csrc. nist. gov/encryption/aes/rijndael/Rijndael. pdf.

[12] FIPS-197. Announcing the ADVANCED ENCRYPTION STANDARD (AES)[S]. 2001. http://csrc, nist. gov/publications/fips/fips 197/fips-197. pdf.

[13] ANSI, X9. 62, 2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)[S], U. S. A, ANSI, 2005.

[14] Certicom Company. http://www. certicom. com/index. php? action = ecc, home.

[15] Cohen H. A Course in Computational Algebraic Number Theory(GTM 138)[M]. 英文版. 北京:世界图书出版公司,1997.

[16] Diffie W, Hellman M. New Directions in Cryptography[J]. IEEE Transactions on Information Theory:1976, IT-22(6):644-654.

[17] Hankerson D, Hernandez J, Menezes A J. Software Implementation of Elliptic Curve Cryptography over Binary Fields[C]. CHES 2000, 2000, Berlin, Hedelberg, Springer-Verlag, LNCS 1965:1-24.

[18] IEEE, IEEE P1363-2000, Standard Specifications For Public-Key Cryptography[S], http://grouper. ieee. org/groups/1363/.

[19] ISO/IEC, ISO/IEC 15946, 2002, Information Technology——Security Techniques—Cryptographic Techniques Based on Elliptic Curves[S], http://www. iso. org/iso/en/ISOOnline. frontpage.

[20] Johnson D, Menezes A J, Vanstone S. The Elliptic Curve Digital Signature Algorithm(ECDSA)[J]. International Journal of Information Security, 2001, LNCS 1615: 36-63.

[21] Koblitz N, Menezes A J, Vanston S. The State of Elliptic Curve Cryptography[J]. Designs, Codes and Cryptography: 2000, 19: 173-193.

[22] Koblitz N. Introduction to Elliptic Curves and Modular Forms(GTM 97)[M]. 英文版,北京:世界图书出版公司,2000.

[23] Miller V. Uses of Elliptic Curves in Cryptography[J]. Advances in Cryptography-CRYPTO'85,1986,Berlin, Hedelberg,Springer-Verlag,LNCS 218:209-232.

[24] Rosing Michael. Implementing Elliptic Curve Cryptography[R]. U. S. A. ,Manning Publicaions Co. ,1998.

[25] Sun Lab. Securing the Web with Elliptic Curve Cryptography[Z]. 2005,http://research. sun. com/projects/crypto.

[26] Schneier B. 应用密码学[M]. 北京:机械工业出版社,2000.

[27] Menezes A J, Oorschot P. Vanstone S. Handbook of Applied Cryptography[M]. U. S. A: CRC Press,1996.

[28] Blake I F, Seroussi G, Smart N. P. Elliptic Curves in Cryptography[M]. U. K. , Cambridge University Press,1999.

[29] 陈恭亮. 信息安全数学基础[M]. 北京:清华大学出版社,2004.

[30] 曾谨言. 量子力学:上、下册[M]. 北京:科学出版社,1989.

[31] Einstein,A,Podolsky,B, Rosen,N. Can quantum-mechanical description of physical reality be considered complete? [J]. Physical Review: 1935,47:777 - 780. Reprinted in Quantum theory and measurement (J. A. Wheeler and W. Z. Zurek,eds),Princeton University Press,1983.

[32] Bennett,C H,Bessette F,Brassard G,Salvail L, Smolin J. Experimental quantum cryptography[J]. Journal of Cryptology: 1992,5(1): 3-28. Preliminary version in Advances in Cryptology-Eurocrypt'90 Proceedings. May 1990,Berlin ,Hedelberg,Springer-Verlag: 253-265.

[33] Bennett C H, Brassard G ,Crépeau C, Skubiszewska M H. Practical quantum oblivious transfer[C]. Advances in Cryptology-Crypto'91 Proceedings. August 1991,Berlin,Hedelberg,Springer-Verlag: 351-366.

[34] Ekert A K. Quantum cryptography based on Bell's theorem[J]. Phy. Rev. Lett: 1991,67 (6): 661-663.

[35] Gisin N. Experimental studies of quantum cryptography in optical fiber communication systems[C]. In Proceedings of SPIE. 1999,3749: 342-343.

[36] Chiangga S, et al. Towards practical quantum cryptography[J]. Applied Physics B: Lasers and Optics, 1999,69 (5) : 389-393.

[37] Maurer U M. Secret key agreement by public discussion from common information[J]. IEEE Transactions on Information Theory: 1993,39(3), 733-742.

[38] 吴祥兴,陈忠. 混沌学导论[M]. 上海:上海科学技术文献出版社. 1996.

[39] 王光瑞,于熙龄,陈式刚. 混沌的控制、同步与应用[M]. 北京:国防工业出版社,2001

[40] 邹恩,李祥飞,陈建国. 混沌控制及其优化应用[M]. 长沙:国防科技大学出版社. 2002.

[41] 王育民,何大可. 保密学——基础与应用[M]. 西安:西安电子科技大学出版社, 1990.

[42] 邓绍江. 混沌理论及其在信息安全中的应用研究[D]. 重庆:重庆大学, 2005.

[43] Tohru Kohda, Akio Tsuneda. Statistics of chaotic binary sequences[J]. IEEE Trans Information Theory, 1997. 43(1): 104-112.

[44] Janusz Szczepnski, Zbigniew Kotulski. Pseudorandom number generators based on chaotic dynamical systems [J]. Open Sys. & Information Dyn: 2001. 8(2): 137-146.

[45] Dedieu H,Kennedy M P,Hasler M. Chaos shift keying: Modulation and demodulation of a chaotic carrier using self syncharonizing Chua's circuits[J]. IEEE Trans. On CAS-2,1993,40(10):634-642.

[46] Kocarev L, et al. General Approcach for Chaotic Synchronization with applications to communication[J]. Phys. Rev. Lett:1995,74(25): 5028-5031.

[47] Shio T U, Inoue E. Chaos communication using observer based chaos synchronization[C]. 14th world congress of IFAC,1999,479 - 484.

[48] Grassi G, Mascolo S. A system theory approach for designing cryptosystems based on hyperchaos[J]. IEEE Transactions on circuits and systems I: 1999,46(9): 1135-1138.

[49] Lee C,Williams D B, Lee J. A secure communications system using chaotic switching[J]. Int. J. Bifur. And Chaos: 1997,7(6): 1383-1394.

[50] 肖国镇,等. 密码学的新领域——DNA 密码[J]. 科学通报: 2006,51(10): 1139-1144.

[51] Adleman L. Molecular computation of solutions to combinatorial problems[J]. Science: 1994(266): 1021-1023.

[52] Sakamoto K,Gouzu H,Komiya K,et al. Molecular computation by DNA hairpin formation[J]. Science:2000 (288): 1223-1226.

[53] Gehani A,LaBean T H,Reif J H. DNA-based cryptography[J]. Dimacs Series in Discrete Mathematics and Theoretical Computer Science:2000(54): 233-249.

[54] Lipton R J. Using DNA to solve NP-complete problems[J]. Science:1995(268): 542-545.

[55] Leiter A,Richter C,Banzhaf W,et al. Cryptography with DNA binary strands[J]. Biosystems:2000,57(1): 13-22.

[56] Sam Roweis et al. A sticker based model for DNA computation[J]. Journal of Computational Biology:1998, 5(4): 615-629.

[57] Willem P C, Stemmer. The evolution of molecular computation[J]. Science: 1995(270): 1510-1510.

[58] Shalon D,Smith S J,Brown P O. A DNA microarray system for analyzing complex DNA samples using two-color fluorescent probe hybridization[J]. Genome Res:1996,6(7): 639-649.

[59] Ravinderjit S, et al. Solution of a 20-variable 3-SAT problem on a DNA computer[J]. Science:2002(266): 499-502.

[60] Bennett C H,Brassard G,Ekert A K. Quantum cryptography[J]. Science: 1992(267): 50-57.